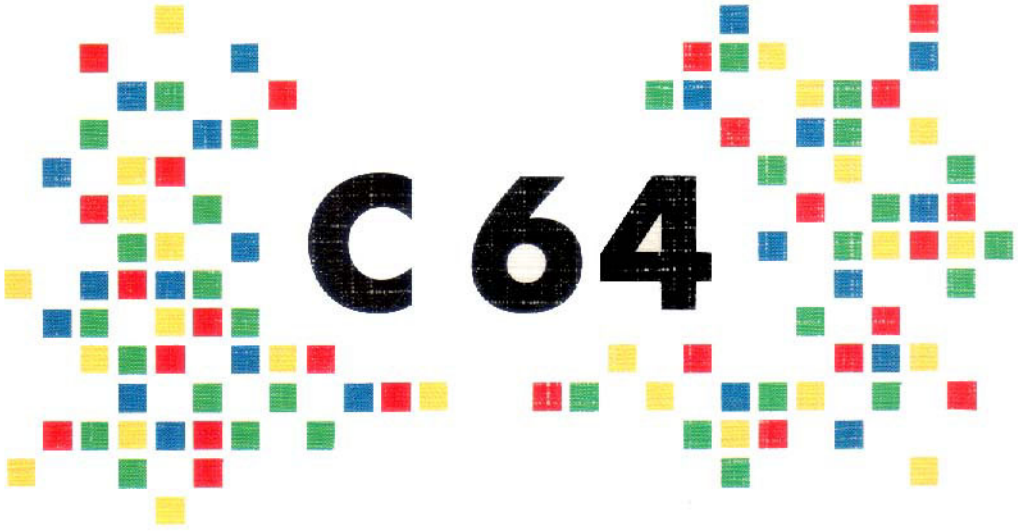


Instrukcja obsługi

Commodore

WSTĘP **4**
ROZDZIAŁ 1 ..... **5**
PRZYGOTOWANIE KOMPUTERA DO PRACY **5**
ROZPAKOWANIE I PODŁĄCZENIE KOMPUTERA 5
INSTALACJA 6
PODŁĄCZENIE DODATKOWYCH URZĄDZEŃ 6
KONTROLA DZIAŁANIA 7
USUWANIE USTEREK 8
USTAWIANIE KOLORÓW 8
ROZSZERZANIE SYSTEMU DODATKOWYMI URZĄDZENIAMI PERYFERYJNYMI 9
ROZDZIAŁ 2 ..... **11**
PIERWSZE KROKI **11**
KOMUNIKACJA Z C 64: KLAWIATURA 11
WCZYTYWANIE PROGRAMÓW 13
FORMATOWANIE NOWYCH DYSKIETEK 14
ZAPISYWANIE PROGRAMÓW 14
WYŚWIETLANIE KATALOGU PROGRAMÓW NA DYSKU 15
ROZDZIAŁ 3 ..... **16**
WSTĘP DO JĘZYKA BASIC **16**
WYŚWIETLANIE TEKSTU I OBLICZENIA 16
FUNKCJE MATEMATYCZNE 17
WIELOKROTNE OBLICZENIA W JEDNEJ LINII 17
KOLEJNOŚĆ WYKONYWANIA DZIAŁAŃ 18
ŁĄCZENIE MOŻLIWOŚCI KOMENDY PRINT 19
ROZDZIAŁ 4 ..... **20**
PISANIE PROSTYCH PROGRAMÓW W JĘZYKU BASIC **20**
NUMERY LINII 20
KOMENDA GOTO 21
KOMENDA LIST 21
WSKAZÓWKI DO EDYCJI 21
JAK KORZYSTAĆ ZE ZMIENNYCH 22
PĘTLA FOR ... NEXT 23
KOMENDA IF/THEN 25
ROZDZIAŁ 5 ..... **26**
SZERSZE INFORMACJE O JĘZYKU BASIC **26**
WSTĘP 26
PROSTA ANIMACJA 26
KOMENDA INPUT 27
UŻYWANIE KOMENDY GET DO WPROWADZANIA DANYCH 28
UŻYWANIE KOMENDY GET DO OBSŁUGI KLAWISZY FUNKCYJNYCH 29
LICZBY LOSOWE I INNE FUNKCJE 29
GRA ZGADYWANKA 30
GRA W KOŚCI 31
LOSOWA GRAFIKA 32
ROZDZIAŁ 6 ..... **33**
KOLORY I GRAFIKA **33**
JAK WYKORZYSTYWAĆ KOLORY I GRAFIKĘ KOMPUTERA 33
WYŚWIETLANIE KOLORÓW 33
KODY CHR$ KOLORÓW 34
JAK KORZYSTAĆ Z PEEK I POKE 35
ZNAKI NA EKRANIE 36
MAPA PAMIĘCI EKRANU 36
MAPA PAMIĘCI KOLORÓW 37
INNY PROGRAM Z ODBIJAJACA SIĘ KULKA 37
ROZDZIAŁ 7 ..... **39**
WPROWADZENIE W GRAFIKĘ SPRITE **39**
BITY I BAJTY 39
TWORZENIE GRAFIKI SPRITE 41
PROJEKTOWANIE OBIEKTU SPRITE 41
WSKAŹNIKI PÓL SPRITE 43
WŁĄCZANIE SPRITE 43
UMIESZCZANIE OBIEKTU SPRITE NA EKRANIE 44

POWIĘKSZANIE OBIEKTÓW SPRITE ... 45
TWORZENIE KILKU OBIEKTÓW ... 45
PRIORYTETY OBIEKTÓW ... 46
WYŁĄCZANIE OBIEKTÓW ... 46
OPERATORY LOGICZNE ... 46
ROZDZIAŁ 8 ... 48
DŹWIĘK I MUZYKA ... 48
UKŁAD SID ... 48
PRZYKŁADOWY PROGRAM MUZYCZNY ... 52
ODTWARZANIE PIOSENEK NA C 64 ... 52
TWORZENIE EFEKTÓW DŹWIĘKOWYCH ... 53
FILTRY ... 54
MUSIC COMPOSER ... 55
ROZDZIAŁ 9 ... 56
ZAAWANSOWANE TECHNIKI OPEROWANIA DANYMI ... 56
KOMENDY READ I DATA ... 56
OBLICZANIE WARTOŚCI ŚREDNIEJ ... 57
TABLICE ... 58
DEFINIOWANIE WIELKOŚCI TABLIC ... 59
SYMULACJA RZUTÓW KOSTKA Z UŻYCIEM TABLIC ... 59
TABLICE DWUWYMIAROWE ... 60
DODATKI ... 63
DODATEK A ... 63
ROZSZERZANIE SYSTEMU KOMPUTEROWEGO ... 63
DODATEK B ... 68
OPIS KOMUNIKATÓW BŁĘDÓW DOS ... 68
DODATEK C ... 70
BASIC COMMODORE 64 ... 70
DODATEK D ... 79
SKRÓTY KOMEND BASIC ... 79
DODATEK E ... 81
KODY EKRANOWE ... 81
DODATEK F ... 83
KODY ASCII I CHR$ ... 83
DODATEK G ... 85
MAPY PAMIĘCI EKRANU I KOLORÓW ... 85
DODATEK H ... 86
WZORY NA FUNKCJE MATEMATYCZNE ... 86
DODATEK I ... 87
UKŁAD WYPROWADZEŃ SYGNAŁÓW W GNIAZDACH KOMPUTERA ... 87
DODATEK J ... 89
PRZYKŁADOWE PROGRAMY ... 89
DODATEK K ... 92
ZAMIANA PROGRAMÓW NAPISANYCH W INNYCH DIALEKTACH JĘZYKA BASIC NA COMMODORE BASIC ... 92
DODATEK L ... 93
KOMUNIKATY BŁĘDÓW ... 93
DODATEK M ... 94
CZĘSTOTLIWOŚCI NUT ... 94
DODATEK N ... 96
LITERATURA ... 96
DODATEK O ... 97
MAPA REJESTRÓW SPRITE ... 97
DODATEK P ... 99
MAPA REJESTRÓW UKŁADU 6566/6567 (VIC II) ... 99
DODATEK Q ... 100
USTAWIANIE PARAMETRÓW DŹWIĘKU W COMMODORE 64 ... 100
DODATEK R ... 102
SPECYFIKACJA UKŁADU 6581 (SID) ... 102
DODATEK S ... 104
KOMENDY OBSŁUGUJĄCE STACJĘ DYSKÓW I DRUKARKĘ ... 104

WSTĘP

Komputer COMMODORE 64 jest najlepszym z obecnie dostępnych komputerów domowych. COMMODORE 64 można wykorzystywać do wszelkich zajęć; od pomocy w prowadzeniu interesów, poprzez pisanie listów do wspaniałych gier. COMMODORE 64 posiada obszerna pamięć (64kB), szeroka gamę kolorów (16), bogate efekty dźwiękowe. Można użytkować na nim gotowe programy lub pisać własne w łatwym do nauki języku programowania BASIC.

Niniejsza instrukcja zawiera informacje potrzebne do prawidłowego podłączenia komputera, zrozumienia jak działa COMMODORE 64, oraz do nauki pisania prostych programów w języku BASIC.

Zadaniem tej instrukcji jest przedstawienie komputera, jednakże nie omawia ona szczegółowo wszystkich informacji o komputerze ani o języku BASIC. W instrukcji podana jest literatura uzupełniająca wyjaśniająca szerzej przedstawiane hasła.

Ci z Państwa, którzy nie mają zamiaru uczyć się programowania w języku BASIC, nie będą musieli przeglądać całej instrukcji, aby nauczyć się podłączać komputer i korzystać z gotowych gier i programów. Informacje zostały opisane w rozdziałach 1 i 2.

COMMODORE 64 został wyposażony w jeden z najbardziej zaawansowanych układów graficznych nazywany grafiką SPRITE. Grafika SPRITE pozwala na:

- Projektowanie własnych kolorowych obiektów graficznych takich, jakie spotyka się w grach komputerowych.
- Animację do 8 obiektów.
- Poruszanie utworzonych obiektów po całym ekranie.
- Dwukrotne powiększenie rozmiarów utworzonych obiektów.
- Zasłanianie jednych obiektów przez inne.
- Korzystanie z automatycznego wykrywania kolizji, dzięki któremu komputer wykonuje to, co chcemy po kolizji dwóch obiektów.

Grafika Sprite pozwala na zaprojektowanie własnych gier.

COMMODORE 64 posiada również wbudowany układ dźwiękowy, który może rywalizować z wieloma dobrze znanymi syntezatorami muzycznymi.

Układ ten zawiera:

- 3 niezależne generatory dźwięku, każdy o zakresie 9 oktaw.
- 4 kształty przebiegu (piłokształtny, trójkątny, prostokątny, szumy).
- Programowalny generator obwiedni.
- Programowalny filtr dolno-, górno- lub pasmowozaporowy.
- Zmienny rezonans i regulacja głośności.

Aby muzyka mogła być odtwarzana z profesjonalną jakością, COMMODORE 64 posiada gniazdo pozwalające podłączyć niemal każdy wzmacniacz.

W miarę coraz pełniejszego wykorzystania komputera można rozszerzać jego możliwości poprzez dołączanie kolejnych elementów wyposażenia zwanych urządzeniami peryferyjnymi. Takimi urządzeniami są:

- Magnetofon DATASSETTE, do korzystania z taśm.
- Stacja dysków VIC 1541 (można dołączyć do pięciu takich stacji).
- Drukarki mozaikowe (igłowe) COMMODORE do wydruku programów, listów, pism itp.
- Przystawka MODEM służąca do połączenia poprzez linię telefoniczną z bazami danych dużych komputerów, z setkami wyspecjalizowanych sieci usługowych oraz z różnymi sieciami informatycznymi.
- Monitor kolorowy COMMODORE 1701.

Posiadacze stacji dysków VIC 1540 mogą przystosować ją do pracy z COMMODORE 64 w punktach serwisowych firmy.

Firma COMMODORE życzy Państwu wiele radości z użytkowania komputera. Jednocześnie zwraca uwagę, że nauka programowania zajmuje dużo czasu. Należy cierpliwie zapoznać się z instrukcją obsługi.

ROZDZIAŁ 1

PRZYGOTOWANIE KOMPUTERA DO PRACY

- **Rozpakowanie i podłączenie komputera**
- **Instalacja**
- **Podłączenie dodatkowych urządzeń**
- **Kontrola działania**
- **Usuwanie usterek**
- **Ustawianie kolorów**
- **Rozszerzanie systemu dodatkowymi urządzeniami peryferyjnymi.**

ROZPAKOWANIE I PODŁĄCZENIE KOMPUTERA

Poniższe informacje wyjaśniają jak podłączyć COMMODORE 64 do telewizora, wzmacniacza lub monitora i sprawdzić czy wszystkie elementy pracują poprawnie.

Przed podłączeniem czegokolwiek do komputera, należy sprawdzić zawartość kartonu, w którym komputer był sprzedany. Poza niniejszą instrukcją, w kartonie powinny znajdować się:

1. Commodore 64
2. Zasilacz (szare pudełko z kablami do sieci i do komputera)
3. Kabel video (do podłączenia telewizora).

W przypadku stwierdzenia braku części należy bezzwłocznie odnieść komputer do sprzedawcy w celu wymiany.

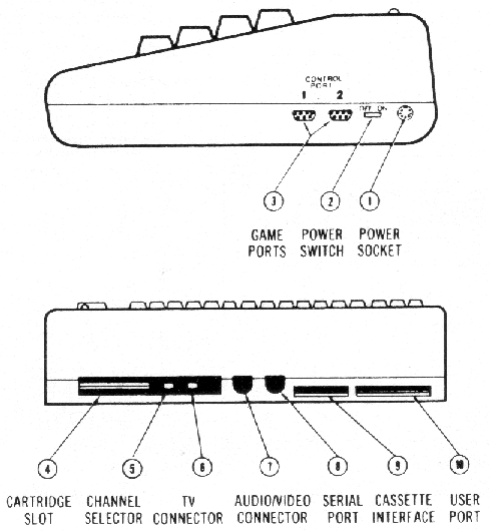
Na początek rozmieszczenie, układ gniazd komputera i ich opis.

GNIAZDA NA BOCZNEJ ŚCIANCE

1. **Gniazdo zasilania.** Do tego gniazda należy podłączyć kabel z zasilacza.
2. **Włącznik zasilania.** Służy do włączania komputera.
3. **Gniazda podłączenia joysticka** (Game ports). Do każdego z tych gniazd można podłączyć joystick lub wiosełka (paddle). Ponadto do gniazda położonego najbliżej przodu komputera można podłączyć pióro świetlne (light pen).

GNIAZDA NA TYLNEJ ŚCIANCE

4. **Gniazdo cartridge'a.** Prostokątne gniazdo, do którego można podłączyć cartrigde z grą lub programem.
5. **Przełącznik kanału.** Przełącznikiem tym wybiera się kanał telewizyjny, na którym wysyłany jest obraz z komputera do telewizora.
6. **Gniazdo TV.** Do tego gniazda podłączany jest kabel do telewizora, przez który wysyłany jest obraz i dźwięk.
7. **Wyjście głosu i wizji** (Audio & Video). Do tego gniazda można podłączyć wzmacniacz fonii oraz telewizor wyposażony w wejście video lub monitor np. Commodore 1701.
8. **Złącze szeregowe** (Serial Port). Do tego gniazda można podłączyć drukarkę COMMODORE lub stację dysków VIC 1541.
9. **Złącze magnetofonu.** Do tego złącza można dołączyć magnetofon DATASSETTE, na który i z którego można wczytywać programy.
10. **User Port.** Do tego złącza można podłączyć dodatkowe urządzenia peryferyjne, takie jak MODEM czy interface komunikacyjny RS-232.



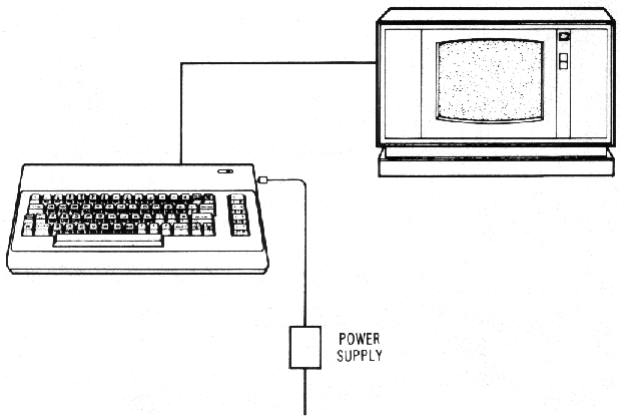
INSTALACJA

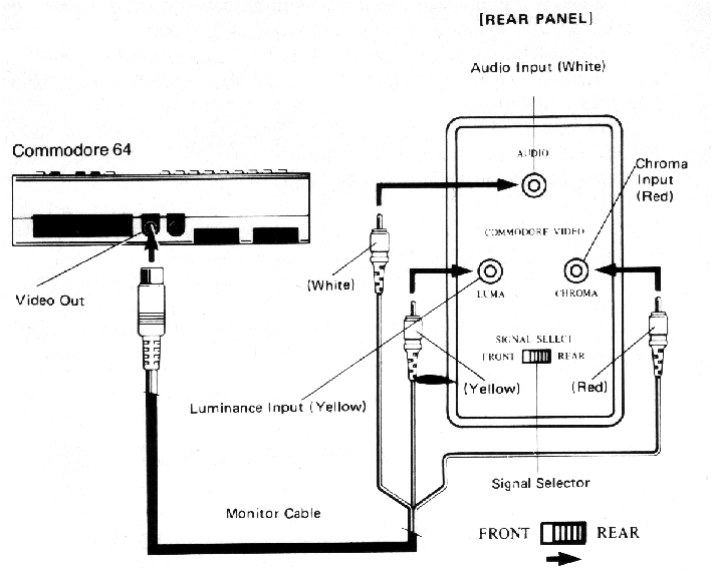
PODŁĄCZENIE DO TELEWIZORA

Komputer należy podłączyć do telewizora zgodnie z rysunkiem poniżej.

1. Podłączyć jeden koniec kabla telewizyjnego do gniazda w C 64.
2. Podłączyć drugi koniec kabla do gniazda antenowego w telewizorze.
3. Podłączyć kabel z zasilacza do gniazda w komputerze. Wtyczka jest tak skonstruowana, że można ją podłączyć tylko w jeden sposób. Zasilacz służy do zamiany napięcia z sieci na napięcia potrzebne do pracy komputera.

Komputer jest już prawidłowo podłączony. Nie potrzeba żadnych dodatkowych podłączeń, aby używać komputer z telewizorem.





PODŁĄCZENIE DODATKOWYCH URZĄDZEŃ

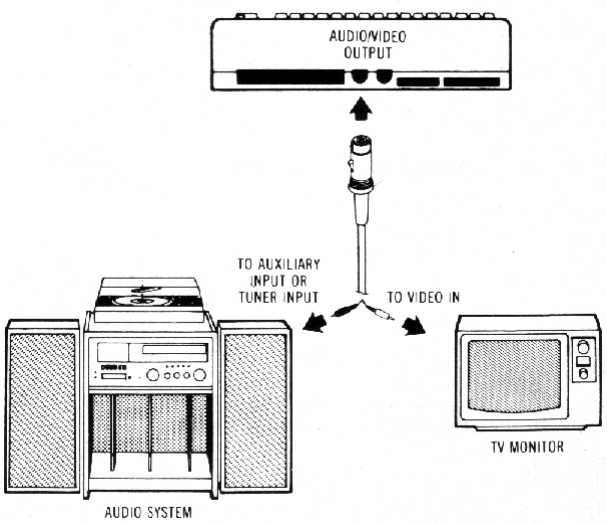
Commodore 64 zawiera kanał dźwięku wysokiej jakości. Istnieje możliwość podłączenia do wzmacniacza w celu uzyskania najlepszego brzmienia dźwięku. Ponadto komputer wysyła standardowy sygnał video (composite video), można go więc podłączyć do telewizorów z wejściem monitorowym lub monitora.

Możliwości te są dostępne dzięki gniazdu wyjścia wizji i fonii (audio / video output) na tylnej ściance C 64. Najprostszym sposobem korzystania z tego gniazda jest użycie standardowego kabla z wtyczką DIN z pięcioma igłami (nie dostarczanego wraz z zestawem). Dwa z czterech wtyków na drugim końcu kabla podłączone są do sygnałów audio i video. Można także zrobić własny kabel, korzystając z opisu podłączeń zawartego w Dodatku I niniejszej instrukcji.

Zazwyczaj CZARNA wtyczka kabla podłączona jest do sygnału AUDIO. Wtyczkę tę należy podłączyć do wejścia AUXILIARY we wzmacniaczu lub AUDIO IN w monitorze lub magnetowidzie.

BIAŁA lub CZERWONA wtyczka zazwyczaj podłączona jest do sygnału VIDEO. Powinna ona być podłączona do wejścia VIDEO IN w monitorze lub magnetowidzie.

Zależnie od producenta kabla połączeniowego, kolory wtyczek mogą być inne. Jeżeli po podłączeniu w wyżej opisany sposób nie można uzyskać dźwięku lub wizji należy znaleźć odpowiednie połączenia korzystając z opisu podłączeń zawartego w Dodatku I niniejszej instrukcji.

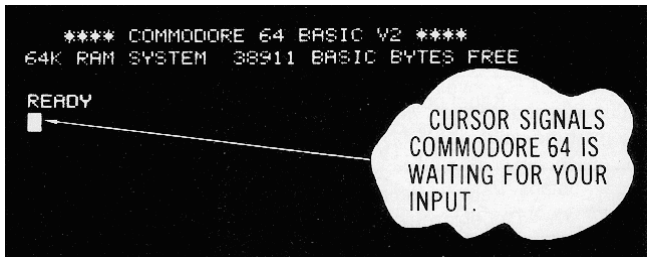


Dodatkowe urządzenia peryferyjne takie jak stacja dysków VIC 1541, drukarki MPS 801,802, 803 plotter 1520 czy monitor 1701 należy podłączać zgodnie ze wskazówkami zawartymi w instrukcjach obsługi tych urządzeń.

Pełen zestaw może wyglądać na przykład tak.

KONTROLA DZIAŁANIA

1. Włączyć komputer włącznikiem znajdującym się na prawej bocznej ściance komputera.
2. Po chwili na ekranie powinien pojawić się następujący obraz.



3. Jeżeli telewizor ma możliwość ręcznego strojenia, dostroić do uzyskania czystego odbioru.
4. Można także regulować nasycenie kolorów aby uzyskać lepszy obraz. Można skorzystać z procedury zmiany kolorów opisanej dalej, aby poprawnie ustawić kolory. Po włączeniu komputera ekran powinien być ciemnoniebieski z jasnoniebieską ramka i literami.

Jeżeli nie udaje się uzyskać spodziewanych rezultatów, należy sprawdzić ponownie połączenia kabli. Poniższe zestawienie pomoże rozwiązać problemy.

USUWANIE USTEREK

Objaw	Przyczyna	Działanie
Nie pali się wskaźnik	Komputer nie włączony	Sprawdź, czy włącznik jest w pozycji "ON"
	Kabel zasilający nie podłączony	Sprawdź, czy wtyczka kabla jest prawidłowo wciśnięta do komputera
	Zasilacz nie podłączony do sieci	Sprawdź podłączenie zasilacza do gniazda w ścianie
	Przepalony bezpiecznik w komputerze	Zanieś komputer do punktu serwisowego w celu wymiany bezpiecznika
Brak obrazu	TV ustawiony na zły kanał	Sprawdź inny kanał (3 lub 4)
	Kabel podłączony do niewłaściwego gniazda w TV	Kabel powinien być podłączony do gniazda VHF
	Kabel video nie podłączony	Sprawdź podłączenie kabla video
	Komputer ustawiony na zły kanał	Przestaw komputer na ten sam kanał co TV (3 lub 4)
Losowe litery na ekranie, założony cartridge	Cartridge nie założony poprawnie	Ponownie wcisnąć cartridge po wyłączeniu zasilania
Obraz bez kolorów	Źle dostrojony TV	Popraw dostrojenie
Słabe kolory	Złe ustawienie kolorów w TV	Popraw ustawienie kolorów
Dźwięk z dużymi szumami	Głośność w TV ustawiona zbyt głośno	Zmień ustawienie głośności
Dobry obraz brak dźwięku	Głośność w TV ustawiona zbyt cicho	Zmień ustawienie głośności
	Wzmacniacz nie podłączony prawidłowo	Sprawdź podłączenie wzmacniacza

Uwagi: COMMODORE 64 został zaprojektowany tak, aby mógł być używany przez każdego.

Jednakże, jak wykazuje doświadczenie naszej firmy, czasami niektórzy użytkownicy komputerów miewają kłopoty z uruchamianiem i obsługą komputerów. Aby pomóc i odpowiedzieć na Państwa pytania, a także podać przykłady prostych programów Commodore wydało szereg publikacji. Ponadto jeżeli działa w pobliżu klub użytkowników Commodore, warto zostać jego członkiem, aby spotykać innych właścicieli C 64, którzy mogą pomóc dzieląc się swoją wiedzą i doświadczeniem.

KURSOR

Pulsujący prostokąt pod napisem READY nazywany jest kursorem. Służy on do oznaczania miejsca na ekranie, gdzie pojawi się napis wpisany na klawiaturze. W trakcie pisania kursor przemieszcza się do przodu o jedno pole, a w miejscu jego poprzedniego położenia pojawia się znak wciśnięty na klawiaturze. Można spróbować wpisać tekst i obserwować kursor przesuwany po ekranie i pojawiające się na ekranie znaki.

USTAWIANIE KOLORÓW

Poniżej podano sposób uzyskania kolorowego obrazu kontrolnego na ekranie, aby móc prawidłowo dostroić kolory w telewizorze. Nawet osoby, które nie są jeszcze zaznajomione z obsługę komputera mogę to zrobić samodzielnie.

Na początek należy znaleźć w lewej części klawiatury klawisz oznaczony CTRL . CTRL jest skrótem od angielskiego słowa

Control, co oznacza "sterowanie". Klawisz ten używany jest w połączeniu z innymi klawiszami aby "powiedzieć" komputerowi, jakie zadanie ma wykonać.

Z funkcji sterujących korzysta się trzymając wciśnięty klawisz CTRL i jednocześnie naciskając inny klawisz.

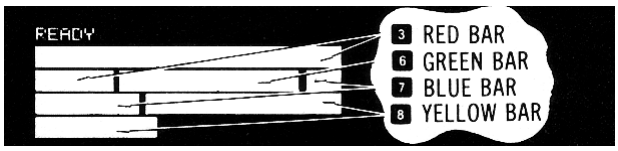
Trzymając wciśnięty klawisz CTRL nacisnąć jednocześnie klawisz 9 . Po zwolnieniu obu klawiszy nic szczególnego się nie wydarzy. Jednakże po naciśnięciu teraz jakiegokolwiek klawisza znak na ekranie ukaże się w odwrotnych kolorach (tzn. znak będzie w kolorze tła, a przestrzeń otaczająca go w kolorze pozostałych znaków).

Następnie należy przytrzymać wciśnięty klawisz spacji (podłużny klawisz na dole klawiatury). Jeżeli powyższa procedura została wykonana prawidłowo na ekranie pojawi się jasno niebieski pasek wydłużający się do końca linii, a następnie przechodzęcy do początku następnej linii, tak długo, jak długo wciśnięty jest klawisz spacji.

Trzymając wciśnięty klawisz CTRL nacisnąć dowolny klawisz z liczbą od 1 do 8. Na przedniej ściance każdego z tych klawiszy wypisany jest kolor, jaki temu klawiszowi odpowiada. Wszystko od tego momentu wyświetlane będzie na ekranie w tym kolorze. Na przykład wciśnięcie klawisza CTRL i jednocześnie klawisza 8 , a następnie, po zwolnieniu obu tych klawiszy, wciśnięcie spacji, spowoduje ukazanie się na ekranie paska w kolorze żółtym. W ten sam sposób można zmieniać kolor na każdy kolejny opisany na klawiszu numerycznym.

Teraz należy zmienić kolor jeszcze kilkakrotnie i za każdym razem wciskać na chwilę spację, a następnie można wyregulować nasycenie kolorów w telewizorze tak, aby obraz był najlepszy.

Ekran powinien wyglądać mniej więcej tak:



Po tych operacjach komputer jest już prawidłowo podłączony. Następne rozdziały poświęcone są nauce programowania w języku BASIC. Jednakże można już teraz korzystać z gotowych gier i programów bez konieczności poznania zasad programowania.

Do każdego programu dołączony jest opis jak należy ten program uruchomić i używać. Zalecane jest jednak, aby zapoznać się z dalszymi rozdziałami tej instrukcji, aby poznać zasady działania komputera.

ROZSZERZANIE SYSTEMU DODATKOWYMI URZĄDZENIAMI PERYFERYJNYMI

Commodore oferuje różne urządzenia peryferyjne, które rozszerzają możliwości wykorzystania komputera. Urządzenia te dzielą się na:

- Urządzenia do zapisu danych
- Drukarki i plottery
- Monitory
- Modemy dla telekomunikacji
- Przystawki do gier
- Moduły mowy i grafiki
- Urządzenia wspomagające edycję tekstów i grafiki

URZĄDZENIA DO ZAPISU DANYCH

Stacje dysków

Stacje dysków Commodore pozwalają na przechowywanie dużych ilości informacji na dyskietkach 5,25 cala. Korzystanie z dyskietek zapewnia szybki zapis i odczyt; nazwy wszystkich zbiorów są automatycznie zapisywane w specjalnym katalogu, skąd mogą być łatwo przeniesione na ekran lub drukarkę.

Ponadto można podłączyć łańcuchowo do komputera kilka stacji dysków. Połączenie łańcuchowe oznacza, że pierwsza stacja dysków jest podłączona bezpośrednio do komputera, druga do pierwszej itd.

Po nabyciu dodatkowej karty rozszerzającej Commodore 64 IEEE można także dołączyć do komputera dowolną stację standardu IEEE jak np. Commodore CBM 8050 lub 4040 Dual Floppy Disk Drive (podwójna stacja dysków).

Rozdział 2 zawiera szersze informacje na temat korzystania ze stacji dysków.

URZĄDZENIA DO DRUKOWANIA I RYSOWANIA

Drukarki

Do C 64 można podłączyć drukarki firmy Commodore. Są to niedrogie drukarki mozaikowe. Po nabyciu dodatkowej karty rozszerzającej Commodore 64 IEEE można także dołączyć do komputera dowolną drukarkę standardu IEEE jak np. Commodore 6400, drukarkę o wysokiej jakości druku (Letter Quality), lub szybką drukarkę mozaikowa 8023.

Printer/Plotter

Printer/Plotter 1520 firmy Commodore ma możliwość drukowania i rysowania w czterech kolorach (czarny, niebieski, czerwony i zielony). Urządzenie to pozwala na kreślenie wykresów liniowych i kołowych oraz skomplikowanych rysunków.

MONITOR 1701/1702

Monitor 14" firmy Commodore zapewnia idealny obraz kolorowy o wysokiej rozdzielczości, co znacznie poprawia komfort pracy z komputerem. Monitor ten może być podłączony do C 64 kablem z wtyczką z 8 igłami. Instrukcja obsługi Monitora 1701/1702, którą dostaje się wraz z kupnem monitora, jasno wyjaśnia sposób podłączenia. Ponadto układ wyprowadzeń sygnałów w gnieździe 8 igłowym można znaleźć w Dodatku I niniejszej instrukcji.

PRZYSTAWKI DO GIER I INNYCH PROGRAMÓW

Firma Commodore oferuje joysticki oraz wiosełka (paddles), które uatrakcyjniają gry komputerowe. Urządzenia te oferowane są również przez inne firmy. Ponadto dostępne jest pióro świetlne firmy Commodore, które wraz z odpowiednim oprogramowaniem pozwala na komunikację z komputerem poprzez wskazanie odpowiedniej funkcji programu na ekranie.

POMOCE GRAFICZNE FIRMY COMMODORE

Firma Commodore dostarcza wiele różnych pomocy do programowania grafiki, włączając w to SIMON'S BASIC, który dodaje 114 nowych komend do BASICa łącznie z komendami graficznymi i ułatwieniami dla programisty, oraz LOGO, łatwy do nauki język programowania z grafiką żółwia.

ROZSZERZENIA MUZYCZNE

Commodore w najbliższej przyszłości będzie oferował klawiaturę muzyczną (Musical Keyboard) oraz przystawkę perkusyjną nazwaną Digi-drum. Oba te produkty będę zawierały specjalne zestawy programów. Przystawki te zwiększą możliwości muzyczne komputerów Commodore 64.

PODŁĄCZANIE DO ZESTAWU STEREO

Możliwości muzyczne Commodore 64 mogę być wzbogacone przez podłączenie komputera do wysokiej jakości wzmacniacza i kolumn stereofonicznych. Kabel z wtykiem 8 igłowym omawiany przy monitorach 1701/1702 może być użyty do podłączenia komputera do wzmacniacza.

PLANOWANIE ZESTAWU KOMPUTEROWEGO DLA WŁASNYCH POTRZEB

Commodore oferuje urządzenia peryferyjne, dzięki którym można stworzyć dostosowany do własnych potrzeb zestaw komputerowy. Firma oferuje różne rodzaje urządzeń do przechowywania danych, drukowania i komunikacji, każdy więc może wybrać to co jest dla niego najlepsze. Więcej informacji na temat urządzeń peryferyjnych można znaleźć w Commodore Peripheral Guide, magazynach poświęconych komputerowi Commodore 64 omówionych w Dodatku R lub w punktach sprzedaży Commodore.

ROZDZIAŁ 2

PIERWSZE KROKI

- **Komunikacja z C 64: Klawiatura**
- **Wczytywanie programów**
- **Formatowanie nowych dyskietek**
- **Zapisywanie programów**
- **Wyświetlanie katalogu programów na dysku**

KOMUNIKACJA Z C 64: KLAWIATURA

Klawiatura komputera pozwala na komunikację z C 64. Używa się jej do przekazywania komputerowi poleceń i do odpowiadania na pytania, które pojawiają się na ekranie.

Klawiatura wygląda jak klawiatura maszyny do pisania, ma jednak dodatkowo kilka klawiszy specjalnych, dzięki którym C 64 ma większe możliwości niż maszyna do pisania. W trakcie czytania następnych kilku stron należy zwrócić szczególną uwagę na klawisze specjalne.

RETURN Klawisz RETURN nakazuje komputerowi analizę właśnie wpisanego tekstu i zapamiętanie go w pamięci. Ponadto klawisz RETURN przesuwa kursor na początek następnej linii.

Uwaga: Pamięć stanowi zbiór wszystkich informacji, z których komputer może na bieżąco korzystać, bez polecenia operatora.

SHIFT Klawisz SHIFT służy do wpisywania dużych liter lub górnych znaków z klawiszy o podwójnym znaczeniu.

Przy korzystaniu z zestawu znaków graficznych, naciśnięcie klawisza łącznie z klawiszem SHIFT powoduje wyświetlenie na ekranie znaku znajdującego się z PRAWEJ strony na przedniej ściance klawisza.

Przy korzystaniu ze specjalnych czterech klawiszy funkcyjnych, naciśnięcie SHIFT powoduje uzyskanie funkcji opisanej na przedniej ściance klawisza (f2, f4, f6 i f8).

KLAWISZE POZWALAJĄCE DOKONYWAĆ POPRAWEK

CRSR Kursor jest to mały prostokąt wskazujący miejsce pracy na ekranie. Do poruszania kursora po ekranie służą dwa klawisze:

CRSR[↑/↓] przesuwa kursor w górę i w dół.

CRSR[←/→] przesuwa kursor w lewo i w prawo.

Naciśnięcie tych klawiszy powoduje przesunięcie kursora odpowiednio w dół i w prawo. Aby przesunąć kursor w górę lub w lewo, nacisnąć klawisz SHIFT i odpowiedni klawisz kursora.

Do przesunięcia kursora o więcej niż jedno pole wystarczy przytrzymać wciśnięty klawisz kursora tak długo, aż kursor znajdzie się w odpowiednim miejscu.

INST/DEL DEL jest skrótem od angielskiego słowa DELETE co oznacza "kasowanie". Naciśnięcie klawisza DEL powoduje cofnięcie kursora o jedną pozycję i skasowanie znajdującego się tam znaku.

```
PRINT "ERROR"#
PRINT "ERROR"
```

Aby usunąć znak znajdujący się w środku zdania, ustawić kursor na polu bezpośrednio po lewej stronie znaku do usunięcia.

```
POPRAW TOP ZDANIE
POPRAW TOPZDANIE
```

Następnie nacisnąć klawisz DEL. Znaki znajdujące się na prawo od usuwanego automatycznie przesuną się o jedno pole w lewo.

POPRAW TO ZDANIE

INST jest skrótem od INSERT co znaczy "wstawianie". Aby wstawić znaki w środku linii, należy trzymając wciśnięty klawisz SHIFT naciskać klawisz INST/DEL.

Jeżeli pominięto jakieś znaki podczas wpisywania tekstu, należy cofnąć kursor w miejsce gdzie mają być wpisane brakujące litery.

BRAK KI LITER
BRAK KI LITER

Następnie trzymając wciśnięty klawisz SHIFT naciskać klawisz INST/DEL, aż do uzyskania wystarczającej ilości miejsca na wpisanie brakujących liter i dopisać te litery. INST nie przesuwa kursora, dodaje tylko puste miejsce między kursorem a tekstem po prawej stronie.

BRAK KI LITER
BRAK KILKU LITER

Korzystając z klawiszy DEL i INST można poprawić błędny fragment tekstu.

COMMODORE 64 TO DOBRY KOMPUTER!
COMMODORE 64 TO KOMPUTER!
COMMODORE 64 TO KOMPUTER!
COMMODORE 64 TO NAJLEPSZY KOMPUTER!

CLR/HOME Klawisz HOME ustawia kursor w lewym górnym rogu ekranu.

Wciśnięcie klawisza CLR/HOME trzymając jednocześnie wciśnięty klawisz SHIFT powoduje wyczyszczenie całego ekranu i ustawienie kursora w górnym lewym rogu.

RESTORE Naciśnięcie klawisza RESTORE powoduje przywrócenie początkowego stanu ekranu i układów wejścia/wyjścia, tzn. wyczyszczenie ekranu, powrót do standardowych kolorów, wyłączenie układów dźwięku i grafiki.

Uwaga: Aby zadziałał klawisz RESTORE należy najpierw przytrzymać wciśnięty klawisz STOP i jednocześnie nacisnąć klawisz RESTORE.

KLAWISZE FUNKCYJNE

Klawisze po prawej stronie klawiatury, oznaczone f1-f8, nazywane są klawiszami funkcyjnymi, i można je zaprogramować do wykonywania wielu różnych zadań. Opis komendy GET, znajdujący się w rozdziale 5, wyjaśnia jak wykorzystywać te klawisze w programach pisanych w języku BASIC.

CTRL Klawisz CTRL pozwala zmieniać kolory oraz wykonywać inne specjalne zadania zwane funkcjami sterującymi.

Aby zmienić kolor należy przytrzymać wciśnięty klawisz CTRL i nacisnąć klawisz z nazwą koloru. Można uzyskać dodatkowych 8 kolorów używając klawisza **C=**. Rozdział 6 zawiera pełniejsze informacje na temat kolorów.

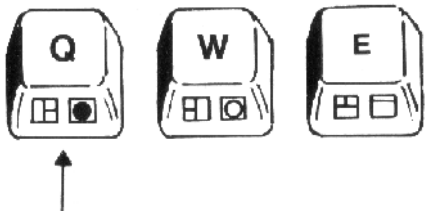
Inne funkcje sterujące uzyskuje się poprzez wciśnięcie klawisza CTRL i naciśnięcie drugiego klawisza. Funkcje sterujące są powszechnie stosowane w gotowych programach, takich jak edytory tekstów.

RUN/STOP Klawiszem STOP można w dowolnej chwili zatrzymać wykonywanie programu napisanego w języku BASIC. Klawiszem tym można także przerwać wyświetlanie tekstu.

Klawisz RUN pozwala wczytać automatycznie program z kasety. Aby użyć funkcji RUN należy użyć tego klawisza łącznie z klawiszem SHIFT.

C= COMMODORE KEY Klawisz **C=** ma dwa zastosowania:

1. **C=** pozwala na przejście z trybu wyświetlania dużych i małych liter (litery i znaki oznaczone na górnej powierzchni klawiszy), na tryb dużych liter i znaków graficznych (duże litery i znaczki narysowane na przednich ściankach klawiszy).
   Aby zmienić tryb pracy, należy nacisnąć jednocześnie klawisze **C=** i S H I FT.
   Po włączeniu komputera ustawiony jest tryb wyświetlania dużych liter i znaków graficznych, co oznacza, że normalne wciśnięcie klawisza powoduje wyświetlenie dużej litery. W tym trybie można wyświetlić na ekranie wszystkie znaki graficzne widoczne na przednich ściankach klawiszy.
   - Aby wyświetlić znaki znajdujące się po prawej stronie należy nacisnąć dany klawisz łącznie z klawiszem SHIFT. Znaki te są wyświetlane tylko w trybie duże litery/grafika.
   - Aby wyświetlić znaki znajdujące się po lewej stronie klawisza, należy nacisnąć klawisz **C=** i jednocześnie dany klawisz. Znaki te wyświetlane są w obu trybach.



2. Klawisz **C=** pozwala również na korzystanie z drugiego zestawu kolorów nie oznaczonego na klawiszach. Aby uzyskać te kolory, należy trzymając klawisz **C=** nacisnąć klawisz z numerem od 1 do 8. W ten sposób powstają następujące kolory:

C= 1 Pomarańczowy	**C=** 5 Szary 2
C= 2 Brązowy	**C=** 6 Jasnozielony
C= 3 Jasnoczerwony	**C=** 7 Jasnoniebieski
C= 4 Szary 1	**C=** 8 Szary 3

WCZYTYWANIE PROGRAMÓW

Commodore 64 pozwala na wczytanie programów z dysku, magnetofonu lub ze specjalnej kasety zwanej cartridge. Oznacza to możliwość korzystania z gotowych programów poprzez proste wczytanie ich do pamięci. A co ważniejsze pozwala także na zapisanie własnych programów na dysk lub magnetofon w celu ponownego ich użycia. Aby użyć ponownie program napisany, a następnie zapisany, wystarczy wczytać do go pamięci i uruchomić.

Przed użyciem stacji dysków lub magnetofonu należy upewnić się, że są one prawidłowo podłączone do komputera.

Dołączanie kasety cartridge

Commodore 64 ma możliwość dołączenia specjalnych kaset z grami lub innymi programami. W takiej formie można dostać szeroką gamę programów i naprawdę ciekawych gier. Aby podłączyć moduł cartridge należy:

1. Wyłączyć komputer.

KOMPUTER ZAWSZE MUSI BYĆ WYŁĄCZONY PODCZAS DOŁĄCZANIA I ODŁĄCZANIA KASETY CARTRIDGE. W PRZECIWNYM WYPADKU ISTNIEJE MOŻLIWOŚĆ USZKODZENIA KASETY I KOMPUTERA.

2. Włożyć kasetę, etykietką do góry do gniazda z tyłu komputera.
3. Włączyć komputer.
4. Rozpocząć grę wciskając klawisz oznaczony jako START na stronie tytułowej gry, która ukaże się na ekranie.

Wczytywanie gotowych programów z kasety magnetofonowej

Gotowe programy są sprzedawane na kasetach magnetofonowych. Są to takie same kasety jakich używa się do nagrywania muzyki.

1. Włożyć kasetę do magnetofonu 1530 DATASSETTE.
2. Upewnić się, że taśma w kasecie jest przewinięta do początku pierwszej strony.
3. Wpisać z klawiatury LOAD i nacisnąć klawisz RETURN. Komputer w odpowiedzi wyświetli napis na ekranie

   PRESS PLAY ON TAPE

4. Wcisnąć klawisz PLAY w magnetofonie. Z ekranu zniknie obraz, aż komputer znajdzie program. Wtedy na ekranie pojawi się napis

   FOUND (NAZWA PROGRAMU)

5. Nacisnąć klawisz **C=**. Obraz z ekranu ponownie zniknie i zacznie wczytywać się program. Wczytywanie programu można w każdej chwili przerwać klawiszem RUN/STOP.

Wczytywanie własnych programów z kasety magnetofonowej

Commodore 64 pozwala zapisać na kasecie swoje własne programy. Jedyne czego do tego potrzeba to magnetofon DATASSETTE 1530 oraz zwykła kaseta magnetofonowa. Aby wczytać program zapisany uprzednio na kasecie należy:

1. Przewinąć taśmę w kasecie do początku.
2. Wpisać z klawiatury LOAD "NAZWA PROGRAMU". Jeżeli nie pamięta się nazwy programu wystarczy wpisać samo LOAD. W ten sposób zostanie wczytany pierwszy programy z taśmy.
3. Nacisnąć klawisz RETURN. Komputer w odpowiedzi wyświetli napis na ekranie
   PRESS PLAY ON TAPE
4. Wcisnąć klawisz PLAY w magnetofonie. Z ekranu zniknie obraz, aż komputer znajdzie program. Wtedy na ekranie pojawi się napis
   FOUND NAZWA PROGRAMU
5. Nacisnąć klawisz **C=**. Obraz z ekranu ponownie zniknie i zacznie wczytywać się program. Po wczytaniu całego programu do pamięci obraz pojawi się na ekranie i wyświetlony zostanie napis READY. Wczytywanie programu można w każdej chwili przerwać klawiszem RUN/STOP.

UWAGA: Po wczytaniu do pamięci nowego programu, wszystkie instrukcje i programy, które były w komputerze zostaną

skasowane. Przed wczytaniem nowego programu, należy upewnić się, że wszystko to, co miało być zachowane zostało zapisane na taśmę lub dysk.

Po wczytaniu programu można go uruchomić poleceniem RUN, wyświetlić jego zawartość poleceniem LIST i dokonać ewentualnych zmian. Jednocześnie należy pamiętać, aby zapisać program ponownie, jeżeli chce się zachować nową wersję programu.

Wczytywanie programów z dysku

Dyski, zwane także dyskietkami, są naprawdę bardzo proste w użyciu. Przewagą dysków nad taśmami jest to, że można znaleźć dane zapisane na dysku dużo szybciej niż na taśmie. Ponadto na dysku mieści się więcej danych niż na taśmie.
Gotowe programy, jak i programy własne zapisane na dysk wczytuje się następująco:

1. Włożyć dyskietkę do stacji dysków. Etykieta dysku powinna być skierowana ku górze. Dyskietkę należy tak wsuwać, aby podłużne wycięcie w jej kopercie było skierowane do przodu. Upewnić się, że dyskietka wsunęła się do końca.
2. Zabezpieczyć dyskietkę, przekręcając w dół dźwignię.
3. Wpisać LOAD "NAZWA PROGRAMU",8. 8 jest kodem oznaczającym stację dysków. Trzeba podać ten kod, aby komputer wiedział, że ma wczytywać program z dysku.

Uwaga: Można wgrać pierwszy program z dysku używając znaku zamiast nazwy programu. Należy wtedy wpisać LOAD "*",8.

4. Nacisnąć klawisz RETURN. Dysk zacznie się obracać, a na ekranie ukażą się napisy:

```
SEARCHING FOR NAZWA PROGRAMU
LOADING
READY
```

5. Gdy na ekranie pokaże się napis READY i pojawi się kursor, wpisać RUN. Program jest wczytany do pamięci i gotowy do użytku.

FORMATOWANIE NOWYCH DYSKIETEK

Zanim użyje się po raz pierwszy nowej, czystej dyskietki, należy ją najpierw sformatować. Formatowanie przygotowuje dyskietkę do pracy, wykonując takie czynności jak podział dysku na bloki. Ponadto podczas formatowania tworzony jest katalog dysku używany potem jako spis wszystkich programów, które zostały zapisane na dysk. NIE NALEŻY formatować dyskietek z gotowymi programami.
Formatuje się tylko nowe dyskietki, a nie dyski, na których zostały zapisane jakieś programy chyba, że chce się skasować wszystkie programy na tym dysku.
Aby sformatować nową dyskietkę stosuje się specjalną wersję komendy OPEN i NEW:

```
OPEN 1,8,15,"N0:<nazwa>,<id>"
```

N0 informuje komputer, że ma być sformatowany dysk w stacji 0. Jeżeli podłączona jest podwójna stacja dysków (przez odpowiedni interface), sformatowana zostanie dyskietka w stacji 0.
Nazwa użyta w poleceniu zostanie zapisana w katalogu jako nazwa danej dyskietki. Nazwa może się składać z maksymalnie 16 dowolnych znaków.
Id czyli identyfikator dysku składa się z dowolnych dwóch znaków. Wskazane jest, aby różne dyski miały różne id.
Gdy zgaśnie wskaźnik w stacji dysków wpisać CLOSE 1, i nacisnąć RETURN.

UWAGA! Formatowanie dysku kasuje wszelkie informacje na nim zawarte. Formatować tylko nowe dyski lub dyski, które mają być całkowicie skasowane. Oto parę przykładów poleceń formatujących dyski:

```
OPEN 1,8,15,"N0:MYFILE,A3"
OPEN 1,8,15,"N0:$RECORDS,02"
```

Teraz, gdy już wiadomo jak formatować dyski, można korzystać z nich do zapisu i odczytu programów z Commodore 64. Dodatek S zawiera więcej informacji o komendzie OPEN.

ZAPISYWANIE PROGRAMÓW

Aby ponownie skorzystać z napisanego programu, należy zapisać go przed wczytaniem nowego. W przeciwnym przypadku program przepadnie.
Po wprowadzeniu zmian w programie już zapisanym, należy zapisać go ponownie, aby zachować nową wersję.
Przy ponownym zapisywaniu programu, nowsza wersja zastępuje wersję poprzednią. Aby zachować obie wersje należy nowej wersji nadać inną nazwę.

Zapisywanie programu na dysk

W celu zapisania programu na dysk należy:

1. Wpisać SAVE "NAZWA PROGRAMU",8. 8 jest kodem oznaczającym stację dysków. Trzeba podać ten kod, aby komputer wiedział, że ma zapisywać program na dysk.
2. Nacisnąć klawisz RETURN. Dyskietka zacznie się obracać, a na ekranie pojawią się następujące komunikaty:

```
SAVING "NAZWA PROGRAMU"
OK
READY
```

Zapisywanie programu na taśmie magnetofonowej

W celu zapisania programu na taśmę należy:

1. Wpisać SAVE "NAZWA PROGRAMU". Nazwa może mieć do 16 znaków długości.
2. Nacisnąć klawisz RETURN. Na ekranie pojawi się komunikat:
   PRESS RECORD AND PLAY ON TAPE
3. Wcisnąć klawisze PLAY i RECORD w magnetofonie DATASSETTE. Obraz zniknie z ekranu i wypełni go tło ramki. Gdy ponownie pokaże się napis READY oznacza to, że program został zapisany.

WYŚWIETLANIE KATALOGU PROGRAMÓW NA DYSKU

Po zapisaniu programu na dysk, komputer automatycznie dopisuje go do katalogu (directory), stanowiącego spis wszystkich programów na dysku.

Istnieje możliwość wyświetlenia spisu programów z danego dysku. Aby zobaczyć ten spis należy:

1. Wpisać LOAD "$",8 i nacisnąć klawisz RETURN. Na ekranie pokażą się następujące komunikaty:

```
SEARCHING FOR $
LOADING
READY
```

2. Wpisać LIST i nacisnąć RETURN.

Nazwy programów na dysku zostaną wyświetlone na ekranie.

ROZDZIAŁ 3

WSTĘP DO JĘZYKA BASIC

- **Wyświetlanie tekstu i obliczenia**
- **Funkcje matematyczne**
- **Wielokrotne obliczenia w jednej linii**
- **Kolejność wykonywania działań**
- **Łączenie możliwości komendy PRINT**

WYŚWIETLANIE TEKSTU I OBLICZENIA

Fragment ten przeznaczony jest głównie dla osób nie znających jeszcze języka BASIC. Uczy on jak wykonać proste działania, jak wyświetlenie tekstu na ekranie, czy wykonywanie prostych obliczeń.

Komenda PRINT nakazuje komputerowi wyświetlić coś na ekranie. Komenda PRINT jest jedną z najbardziej przydatnych i użytecznych komend języka BASIC. Można używać jej do wyświetlania niemal wszystkiego, włączając symbole graficzne i wyniki obliczeń. Komendę PRINT używa się w następujący sposób:

1. Wpisać słowo PRINT. Informuje ono komputer, jaki rodzaj działania ma on wykonać.
2. Wpisać cudzysłów ("). Informuje to komputer gdzie zaczyna się tekst, który ma być wyświetlony.
3. Wpisać dowolny tekst, który ma być wyświetlony.
4. Wpisać kończący cudzysłów. Informuje to komputer gdzie kończy się tekst, który ma być wyświetlony.
5. Nacisnąć klawisz RETURN. Nakazuje on komputerowi wykonać instrukcję, którą w tym wypadku jest polecenie wyświetlenia na ekranie napisu.

Na ekranie pojawi się żądany tekst i pojawi się napis READY. Wygląda to tak:

```
PRINT "I LOVE MY COMMODORE"        To trzeba wpisać i nacisnąć RETURN.
I LOVE MY COMMODORE                To pojawi się na ekranie
READY
```

Komputer wyświetli dowolny napis ujęty w cudzysłów. Należy pamiętać, aby wpisać oba znaki cudzysłowu. Jeżeli popełni się błąd podczas wpisywania komendy PRINT, poprawia się korzystając z klawisza INST/DEL. Można poprawiać tekst w dowolny sposób przed naciśnięciem klawisza RETURN.

Jeżeli komenda była wpisana błędnie i naciśnięto klawisz RETURN, komputer nie zrozumie polecenia i wypisze komunikat o błędzie, który pomaga zorientować się, co było nieprawidłowo wprowadzone. Może to wyglądać np. tak:

?SYNTAX ERROR

Jeżeli pojawi się taki komunikat należy sprawdzić gdzie został popełniony błąd. Komputer jest bardzo dokładny i nie rozumie instrukcji zawierających błędy w pisowni. Aby uniknąć błędów należy zwracać uwagę na poprawną formę wpisanych komend.

Najlepszym sposobem na poznanie języka BASIC i komputera jest próbowanie różnych komend i obserwowanie reakcji na nie.

UŻYWANIE KOMENDY PRINT DO OBLICZEŃ

Komenda PRINT może więcej niż tylko wyświetlać na ekranie tekst ujęty w cudzysłów. Można korzystać z niej również do wykonania obliczeń i automatycznego wyświetlenia na ekranie rezultatu tych obliczeń.

1. Wpisać PRINT.
2. Wpisać działanie, które ma być obliczone. NIE wpisywać cudzysłowów.
3. Nacisnąć klawisz RETURN. Komputer wyświetli wynik działanie i napis READY.

Przykład:

```
PRINT 12 + 12          To trzeba wpisać i nacisnąć RETURN.

24                     Komputer wyświetli napis.
READY
```

Należy pamiętać, żeby nie umieszczać w cudzysłowiu działań, które mają być rozwiązane. Jeżeli działanie ujęte jest w cudzysłów komputer rozumie, że ma wyświetlić je, a nie rozwiązać. Na przykład:

```
PRINT "12 + 12"        To trzeba wpisać i nacisnąć RETURN.
```

12 + 12 Komputer wyświetli napis, który był ujęty w cudzysłów.

Tak więc, aby wykorzystywać komendę PRINT jako kalkulator, wystarczy pominąć cudzysłów. Komenda PRINT może służyć do dodawania, odejmowania, mnożenia i dzielenia, podnoszenia do potęgi oraz innych obliczeń matematycznych, jak obliczanie pierwiastka kwadratowego.

FUNKCJE MATEMATYCZNE

DODAWANIE

Aby nakazać komputerowi dodawanie liczb należy użyć znaku plus (+). Należy pamiętać o naciśnięciu klawisza RETURN po wpisaniu komendy PRINT i działania. W ten sposób informuje się komputer, że ma wykonać instrukcję.

ODEJMOWANIE

Do odejmowania używać znaku minus (-). Nacisnąć klawisz RETURN po wpisaniu działania. Na przykład:

```
PRINT 12 – 9        Wpisać i nacisnąć RETURN
3                   Komputer wyświetli wynik.
```

MNOŻENIE

Do mnożenia używać gwiazdki (*). Nie można używać zwyczajnego x ponieważ komputer potraktuje to jako literę x, a nie znak mnożenia. Nacisnąć klawisz RETURN po wpisaniu działania. Na przykład:

```
PRINT 12 * 12       Wpisać i nacisnąć RETURN
144                 Komputer wyświetli wynik.
```

DZIELENIE

Do dzielenia używać znaku (/). Nacisnąć klawisz RETURN po wpisaniu działania. Na przykład:

```
PRINT 144/12        Wpisać i nacisnąć RETURN
12                  Komputer wyświetli wynik.
```

POTĘGOWANIE

Do potęgowania używać strzałki w górę (^). Nacisnąć klawisz RETURN po wpisaniu działania. Na przykład, aby policzyć 12 do 5 potęgi należy wpisać:

```
PRINT 12 ^ 5        Wpisać i nacisnąć RETURN
248832              Komputer wyświetli wynik.
```

Jest to równoważne z działaniem

```
PRINT 12*12*12*12*12
248832
```

Uwaga: W języku BASIC stosowane są skróty w celu przyspieszenia pisania programów. Jeden skrót zastępuje jedną komendę języka BASIC. Na przykład, zamiast słowa PRINT można pisać ?. W dalszej części tej instrukcji podane będę skróty innych komend języka BASIC. Dodatek D zawiera pełną listę tych skrótów i pokazuje jak wygląda na ekranie dana komenda w formie skróconej.

WIELOKROTNE OBLICZENIA W JEDNEJ LINII

Ostatni przykład pokazał, że można wykonać więcej niż jedno obliczenie w linii. Można również wykonywać różne rodzaje obliczeń w jednej linii. Na przykład:

```
? 3 * 5 - 7 + 2     Wpisać i nacisnąć RETURN
10                  Komputer wyświetli wynik.
```

Jak dotąd w przykładach używane były małe liczby i proste zadania. Commodore 64 potrafi także liczyć bardziej skomplikowane działania. Następne działanie operuje dużymi liczbami.

Należy zwrócić uwagę, że w liczbie 78956.87 nie występuje żaden znak oddzielający między 8 a 9, a część ułamkowa oddzielona jest znakiem kropki. W języku BASIC nie można używać przecinka do tego celu. Język BASIC zakłada, że przecinek oznacza następną liczbę, tak więc zapis 78956,87 potraktowałby jako dwie liczby: 78956 i 87. Należy pamiętać o naciśnięciu klawisza RETURN po wpisaniu działania.

```
? 1234.5 + 3457.8 + 78956.87
83649.17
```

Commodore 64 może pracować na liczbach do 10 cyfr, ale wyświetla tylko dziewięć cyfr wyniku. Tak więc liczby, które mają ponad 9 cyfr są zaokrąglane. Jeżeli ostatnia cyfra wynosi 5 lub więcej, to zaokrąglana jest w górę; liczby mniejsze od pięciu są zaokrąglane w dół. Oznacza to, że liczba 12123123.45 zostanie zaokrąglona do 12123123.5. Z powodu tego zaokrąglenia wynik podany przez komputer różni się od wyniku otrzymanego przy ręcznym obliczaniu tego przykładu. W tym wypadku prawidłowy wynik wynosi 12131364.817. Widać różnicę wynikającą z zaokrąglenia.

```
? 12123123.45 + 345.78 + 7895.687
12131364.9
```

Commodore 64 wyświetla liczby między 0.01 a 999999999 używając standardowego sposobu zapisu. Liczby spoza tego zakresu są wyświetlane w zapisie naukowym (scientific notation). Zapis naukowy pozwala na zapisanie bardzo dużych lub bardzo małych liczb jako potęgę liczby 10. Na przykład:

```
? 123000000000000000
1.23E+17
```

Inny sposób zapisania tej liczby to: 1.23 * 10 ^ 17. Commodore 64 używa zapisu naukowego dla liczb z wieloma znakami, aby prościej było je przeczytać.

Istnieją ograniczenia dotyczące wielkości liczb na jakich może pracować C 64, nawet korzystając z zapisu naukowego. Te ograniczenia to:

Największa liczba: +/- 1.70141183E+38
Najmniejsza liczba: +/- 2.93873588E-39

KOLEJNOŚĆ WYKONYWANIA DZIAŁAŃ

Przy próbach wykonywania działań mogło się zdarzyć, że wynik podany przez komputer różnił się od spodziewanego. Wynika to z tego, że komputer wykonuje obliczenia w pewnej kolejności. Przy obliczaniu:

```
20+8 / 2
```

wynik wynosi 14 jeżeli najpierw doda się 20 i 8, a następnie wynik podzieli przez 2. Ale jeżeli najpierw podzieli się 8 przez 2, a potem doda 20, to wynik wyniesie 24.

Commodore 64 zawsze poda wynik 24 ponieważ komputer wykonuje działania w określonej kolejności. Działania są rozwiązywane od lewej do prawej, ale poza tym generalnym kierunkiem, różne działania mają różne priorytety.

Oto kolejność tych działań:

Pierwsze	**-**	znak minus przed liczbami ujemnymi, nie znak odejmowania,
Drugie	**^**	potęgowanie, od lewej do prawej,
Trzecie	*** /**	mnożenie i dzielenie, od lewej do prawej,
Czwarte	**+ -**	dodawanie i odejmowanie, od lewej do prawej.

Oznacza to, że komputer najpierw szuka w całym działaniu liczb ujemnych, zanim rozpocznie obliczenia. Następnie szuka potęgowania, potem wykonuje wszystkie mnożenia i dzielenia, a dopiero na koniec dodaje i odejmuje.

To wyjaśnia dlaczego 20 + 8 / 2 wynosi 24: 8 jest dzielone przez 2 zanim zostanie dodane 20 ponieważ dzielenie ma priorytet przed dodawaniem.

Istnieje bardzo prosty sposób na wymuszenie innej kolejności działań: wyrażenia, które mają być policzone najpierw należy ująć w nawiasy. Jeżeli do powyższego wyrażenia dodać nawiasy to:

```
? (20 + 8) / 2
```

wynik wynosi 14, ponieważ dzięki nawiasom komputer najpierw dodał 20 i 8, a dopiero potem podzielił przez 2.

Oto kilka następnych przykładów, jak zmienić kolejność obliczeń przez użycie nawiasów:

```
? 30 + 15 * 2 - 3
57

? (30 + 15) * 2 - 3
87

? 30 + 15 * (2 – 3)
15
```

```
? (30 + 15) * (2 - 3)
-45
```

Ostatni przykład zawiera dwa wyrażenia w nawiasach. Jak zwykle obliczane są one od lewej do prawej, a następnie obliczany jest łączny wynik. Jeżeli występuje więcej niż jedno wyrażenie w nawiasie, można dalej wymusić kolejność obliczeń używając nawiasów wewnątrz nawiasów. Jako pierwsze obliczane jest wyrażenie w nawiasach wewnętrznych. np.:

```
? 30 + (15 * (2 - 3))
15
```

W takim wypadku najpierw odejmowane jest 3 od 2, następnie 15 jest mnożone razy -1, potem -15 jest dodawane do 30. Można eksperymentować z rozwiązywaniem takich wyrażeń, aby zaznajomić się dokładnie z kolejnością, w jakiej wykonywane są działania.

ŁĄCZENIE MOŻLIWOŚCI KOMENDY PRINT

C 64 pozwala na jednoczesne używanie obu form komendy PRINT, o których była mowa uprzednio. Należy zapamiętać, że wszystko, co jest ujęte w cudzysłów będzie wyświetlone tak, jak zostało zapisane.

Następny przykład pokazuje jak można łączyć obie formy komendy PRINT. Wyrażenie ujęte w cudzysłów jest wyświetlane na ekranie bez obliczenia, wyrażenie bez cudzysłowu jest obliczane. Średnik oddziela poszczególne wyrażenia w komendzie PRINT. (Średnik oznacza wypisanie bez odstępu).

```
?  "5 * 9 = " ; 5 * 9          Wpisać i nacisnąć RETURN
5 * 9 = 45                     Komputer wypisze na ekranie.
```

Należy zapamiętać, że tylko drugie wyrażenie jest faktycznie obliczane. Dwa wyrażenia są oddzielone średnikiem. Kolejne wyrażenia po komendzie PRINT muszę być od siebie oddzielone średnikiem lub przecinkiem. Jeżeli zamiast średnika użyje się przecinka to, po wyświetleniu na ekranie, między kolejnymi wyrażeniami będzie wolna przestrzeń. Użycie średnika nie pozostawia wolnej przestrzeni.

Ekran w C 64 zorganizowany jest w 4 strefy po 10 kolumn. Użycie przecinka do oddzielenia kolejnych wyrażeń w komendzie PRINT powoduje wyświetlenie kolejnych wyrażeń w kolejnych strefach. Np:

```
?  "total:" ; 95, "shortage:" ; 15
total:95 shortage:15
```

Jeżeli jest więcej niż cztery wyrażenia, wyświetlanie automatycznie przejdzie do następnej linii. Na przykład:

```
? 2 * 3, 4 - 6, 2 ^ 3, 6/4, 100 + (-48)
6     -2     8     1.5
52
```

Jeżeli użyje się średników:

```
? 2 * 3; 4 - 6; 2 ^ 3; 6/4; 100 + (-48)
6-2  8  1.5  52
```

Różnicę w wyświetlaniu przy użyciu średnika i przecinka można wykorzystywać do tworzenia bardziej skomplikowanych formatów wyświetlania.

ROZDZIAŁ 4

PISANIE PROSTYCH PROGRAMÓW W JĘZYKU BASIC

- **Numery linii**
- **Komenda GOTO**
- **Komendy LIST**
- **Wskazówki do edycji**
- **Jak korzystać ze zmiennych**
- **Pętla FOR ... NEXT**
- **Komenda IF/THEN**

Jak dotąd wyjaśniono w tej książce jak wykonywać proste czynności na komputerze C 64. Wpisywane były pojedyncze linie instrukcji powodującą natychmiastową reakcję komputera, zaraz po naciśnięciu klawisza RETURN. Taki sposób pracy z komputerem nazywany jest trybem bezpośrednim lub kalkulatorowym.

Zazwyczaj wykorzystuje się komputer do wykonywania bardziej złożonych zadań, które korzystają z więcej niż jednego polecenia. Łącząc większą liczbę poleceń w PROGRAM można wykorzystać całą moc komputera.

Żeby zobaczyć jak łatwo jest napisać swój pierwszy program wystarczy wykonaj kolejne punkty:

1. Wyczyścić ekran, trzymając wciśnięty klawisz SHIFT i naciskając jednocześnie klawisz CLR/HOME.
2. Wpisać NEW i nacisnąć klawisz RETURN. Ta komenda powoduje usunięcie z pamięci wszystkich informacji, które mogły tam pozostać po próbach korzystania z komputera.
3. Przepisać dokładnie poniższe dwie linie:

```
10 ? "COMMODORE 64"
20 GOTO 10
```

4. Należy pamiętać, aby po każdej linii wcisnąć klawisz RETURN. Przy wpisywaniu pierwszej linii, można zauważyć, że komputer nie reaguje na komendę PRINT tak, jak to robił w poprzednich ćwiczeniach. Dzieje się tak, ponieważ linia rozpoczyna się od numeru linii (10). Kiedy podaje się na początku numer linii, komputer wie, że wpisywany jest program, więc czeka na wpisanie całego programu zanim zacznie wykonywać poszczególne instrukcje.
5. Wpisać RUN i nacisnąć klawisz RETURN. Komenda RUN informuje komputer, że program już został wpisany i należy rozpocząć jego wykonywanie. Po uruchomieniu programu ekran będzie wyglądał następująco:

```
COMMODORE 64
COMMODORE 64
COMMODORE 64
COMMODORE 64
COMMODORE 64
COMMODORE 64
```

6. Przerwać wykonywanie programu naciskając klawisz RUN/STOP. Komputer wykonuje polecenia wypisując cały czas COMMODORE 64 tak długo, aż zostanie naciśnięty klawisz RUN/STOP. A tak będzie wyglądał ekran po naciśnięciu klawisza STOP:

```
COMMODORE 64
COMMODORE 64
COMMODORE 64
COMMODORE 64
COMMODORE 64
BREAK IN 10
READY
```

Ten prosty program przedstawia szereg istotnych elementów, które są podstawą pisania wszelkich programów.

NUMERY LINII

Jak wspomniano powyżej w punkcie 4 numer linii przekazuje komputerowi informację o tym, że pisany jest program. Ponadto numer ten określa w jakiej kolejności poszczególne polecenia mają być wykonywane. Gdyby numery linii nie przekazywały tej informacji, komputer nie mógłby wiedzieć, co ma robić najpierw.

Im dłuższy i bardziej złożony jest program, tym ważniejsze jest pamiętać, że komputer całkowicie polega na piszącym program w sprawie KIEDY wykonać dane polecenie i JAKIE to jest polecenie. Zaletę numeracji linii jest, że można wpisać najpierw linię 20, a potem linię 10, ponieważ komputer sprawdza numery linii już wpisanych i umieszcza je we właściwej kolejności do wykonania. Komputer nie sprawdza kolejności w jakiej kolejne linie pojawiają się na ekranie.

Następną zaletę numerów linii jest możliwość użycia wskaźnika danej linii w instrukcji. Jeżeli zachodzi potrzeba cofnięcia

się i powtórzenia wykonania danego polecenia wystarczy wskazać jego numer linii w komendzie GOTO, tak jak w powyższym przykładzie.

KOMENDA GOTO

Po rozpoczęciu wykonywania powyższego przykładowego programu napis COMMODORE 64 był wypisywany na ekranie wielokrotnie, a nie tylko raz, ponieważ użyto komendy GOTO.

Komenda GOTO nakazuje komputerowi przejście do wykonywania określonej linii programu i wykonywanie po kolei instrukcji począwszy od tej linii.

Komendę GOTO można użyć, aby ponownie wykonać daną część programu. Można także nakazać komputerowi, aby opuścił kilka linii, nawet gdyby to oznaczało, że pewne linie nie będą wykonane.

W powyższym przykładzie, komputer wykonywał linię 10 programu (wyświetlał na ekranie napis COMMODORE 64), a następnie przechodził do linii 20. Komenda GOTO w tej linii nakazywała ponowne przejście do linii 10 i wykonanie instrukcji z tej linii. Więc komputer ponownie wyświetlał napis na ekranie po czym przechodził do linii 20, która odsyłała go znów do linii 10 itd.

Takie powtarzanie jednej lub kilku instrukcji nazywane jest PĘTLĄ. Ponieważ w powyższym przykładzie nie ma możliwości wyjścia z tej pętli, program będzie stale wykonywany. Aby zatrzymać pracę komputera trzeba nacisnąć klawisz RUN/STOP. Najlepiej byłoby umieścić w programie instrukcję, która pozwoliłaby przerwać wykonywanie programu bez konieczności naciskania klawisza RUN/STOP. Więcej informacji na temat kończenia pętli znajduje się w dalszej części rozdziału.

KOMENDA LIST

Teraz, gdy wykonywanie programu zostało przerwane, wpisać komendę LIST i nacisnąć klawisz RETURN. Program zostanie wyświetlony na ekranie pomimo, że jego wykonywanie zostało przerwane, ponieważ nadal jest on zapisany w pamięci komputera. Jedyną różnicę jest to, że komputer zmienił wprowadzony znak ? na słowo PRINT. Nie ma to wpływu na działanie programu, po prostu komputer zawsze rozwija skróty komend do ich pełnej formy. Po użyciu komendy LIST komputer zawsze wypisuje linie programu w odpowiedniej kolejności, niezależnie od kolejności, w jakiej zostały one wpisane.

Jedną z zasadniczych różnic pomiędzy pisaniem programu, a wprowadzaniem poszczególnych komend w trybie bezpośrednim (kalkulatorowym), jest to, że po wyczyszczeniu ekranu komendy bezpośrednie przepadają, natomiast, do chwili wczytania następnego programu, program można odtworzyć wpisując po prostu komendę LIST. Można teraz też wprowadzić zmiany w programie, zapisać go na dysk lub magnetofon komendą SAVE lub ponownie uruchomić komendą RUN.

WSKAZÓWKI DO EDYCJI

Jeżeli w trakcie wpisywania linii popełniono jakieś błędy lub, gdy chce się zmienić daną linię, C 64 oferuje możliwość łatwej korekty wpisanego programu.

1. Linię z danym numerem można wpisywać dowolną ilość razy, a komputer automatycznie wymienia starą linię na nową. Aby więc zmienić polecenie w danej linii wystarczy wpisać nową linię z tym samym numerem. Na przykład:

```
10 ? "NA IMIE MAM JAN"
20 ? "MAM 17 LAT"
20 ? "MIESZKAM W WARSZAWIE"
RUN
NA IMIE MAM JAN
MIESZKAM W WARSZAWIE
```

Jak widać pierwsza linia z numerem 20 nie została wykonana, ponieważ została zastąpiona drugą linią 20. Po wpisaniu komendy LIST widać, że tylko druga linia 20 znajduje się w programie.

2. Można łatwo usunąć zbędną linię z programu wpisując tylko jej numer i naciskając klawisz RETURN. Po wpisaniu komendy LIST widać, że dana linia, jak i jej numer nie występują już w programie.
3. Można edytować w łatwy sposób istniejącą już linię. W tym celu należy, używając klawiszy CRSR, ustawić kursor w linii, która ma być poprawiana i dokonać niezbędnych poprawek. Po naciśnięciu klawisza RETURN poprawiona linia zastąpi starą w programie. Do edycji można wykorzystywać klawisz INST/DEL.

Po zakończeniu edycji można sprawdzić poprawki wpisując komendę LIST. Należy pamiętać, że komenda LIST spowoduje wypisanie programu w kolejności zgodnej z numeracją linii, niezależnie od kolejności ich wpisywania.

Można dokonać poprawek w pierwszym, przykładowym programie, usunięcie liczby 64 i dodanie na końcu linii 10 średnika. Po wpisaniu poprawek, upewnić się, że kursor znajduje się poniżej linii 20, zanim wpisze się komendę RUN. A tak teraz działa program:

```
LIST
10 PRINT "COMMODORE ";
20 GOTO 10
READY
RUN
COMMODORE COMMODORE COMMODORE COMMODORE
COMMODORE COMMODORE COMMODORE COMMODORE
BREAK IN 10
READY
```

JAK KORZYSTAĆ ZE ZMIENNYCH

Zmienna jest to symbol, zastępujący daną wartość. Czasami wartość zmiennej nie jest znana przed rozpoczęciem wykonywania programu. Jednym z celów programu może być znalezienie jednej lub więcej wartości zmiennej. Oto przykładowa linia z programu:

```
20 LET X = 28+Y
```

Znak = oznacza "przyjmuje wartość". Komenda LET nie jest konieczna i można ją pominąć.

W tym wyrażeniu X i Y są zmiennymi. Załóżmy, że X oznacza liczbę dni w miesiącu. Zaletą korzystania ze zmiennych jest fakt, że można używać jej wielokrotnie w programie, więc X może oznaczać ilość dni w każdym miesiącu, a nie tylko w jednym z góry określonym. Tutaj z pomocą przychodzi zmienna Y. Każdy miesiąc ma co najmniej 28 dni, więc Y oznacza ilość dni ponad 28. Dalej opisany jest program znajdujący wartości obu tych zmiennych.

Najważniejsze w tej chwili jest zrozumienie jak należy korzystać ze zmiennych, ponieważ ich użycie w programach pozwala na wykonywanie bardziej złożonych zadań. Zmienne pozwalają także na pisanie bardziej uniwersalnych programów.

Wyobraźmy sobie, że komputer posiada szereg szufladek. Podczas pisania programu można wykorzystywać niektóre z tych szufladek do przechowywania pewnych wartości. Należy nadać kolejnym szufladkom nazwy i w trakcie wykonywania programu umieszczać w nich wartości korzystając z tych nazw. Na przykład w powyższym wyrażeniu wykorzystywane są dwie takie szufladki nazwane X i Y. Na początku wykonywania programu szufladki te mają nazwy lecz są puste. Można umieścić wartość w danej szufladce. Np.:

X	Y				
	3				

Teraz zmienna Y ma wartość 3. Zmiennej można przypisać dowolną wartość umieszczając w programie prostą instrukcję:

```
10 Y = 3
```

Ponieważ X równa się 28 plus Y, to w trakcie wykonywania programu do szufladki oznaczonej X również zostanie wpisana wartość.

X	Y				
31	3				

Tak na przykład może wyglądać program:

```
10 Y = 3
20 X = 28+Y
30 ? "LICZBA DNI W MAJU WYNOSI"; X
RUN LICZBA DNI W MAJU WYNOSI 31
```

A oto następny program korzystający ze zmiennych:

```
10 X% = 15
20 X = 23.5
30 X$ = "TOTAL : "
40 Y = X%+X
50 ? X$;Y
```

Po wpisaniu komendy RUN i po wykonaniu linii 30 te szufladki będę miały postać:

X%	X	X$	Y		
15	23.5	TOTAL :			

Po zakończeniu programu Y będzie miało wartość 38.5.

Powyższy przykład używa trzech typów zmiennych:

TYP	SYMBOL	OPIS	PRZYKŁAD	PRZYKŁADOWE WARTOŚCI
Całkowite	%	Liczby całkowite	X%, A1%	15, 102, -3
Łańcuchowe	$	Znaki ujęte w cudzysłów	X$, AB$	”TOTAL:”, ”DAY 1”
Zmiennoprzecinkowe		Liczby rzeczywiste i całkowite	X, AB	23.5, 12, 1.3E+2

Należy zwracać uwagę na używanie odpowiednich typów zmiennych w programie. Przy próbie przyporządkowania łańcucha znaków zmiennej całkowitej, program nie będzie działał.
Kilka dodatkowych uwag dotyczących nazw zmiennych:

- Nazwa zmiennej może składać się z jednego lub dwóch znaków, nie licząc specjalnego znaku używanego przy zmiennych całkowitych i łańcuchowych.
- Można używać nazw zmiennych dłuższych niż dwa znaki, ale komputer rozpoznaje tylko dwa pierwsze. Tak więc komputer uzna zmienne np. PA, PARTNO, PAGENO, jako jedną zmienną odnoszącą się do jednej "szufladki".
- Program jest bardziej zrozumiały, gdy używa się dłuższych nazw zmiennych, należy jednak zwracać uwagę na to, aby dwie pierwsze litery nazwy były różne w poszczególnych zmiennych.
- Można w jednym programie używać zmiennych X, X% i X$ ponieważ symbole % i $ powodują, że komputer traktuje je jako osobne zmienne. Mogą to być np. zmienne A2, A2% i A2$.
- Pierwszy znak nazwy zmiennej musi być literą (od A do Z). Drugi może być literą lub cyfrą (0 do 9). Należy pamiętać, że komputer ignoruje następne znaki poza znakiem $ lub % na trzeciej pozycji.
- Nazwy nie mogą zawierać nazw komend i funkcji języka BASIC, nazywanych słowami kluczowymi lub zarezerwowanymi, takich jak RUN czy PRINT, będących częścią języka BASIC. Dodatek D zawiera listę słów kluczowych języka BASIC.

Oto następny program przykładowy pokazujący jak wykorzystywać zmienne w programach. Program ten zawiera ponadto kilka innych komend, o których była dotąd mowa.

```
NEW
10 X = 1.05
20 Y = 300
30 Z =X * Y
40 PRINT "SEATS AVAILABLE:";Y
50 PRINT "TICKETS AVAILABLE:";Z
60 Y = Y + 1
70 PRINT "OVERBOOKING POINT:";Y
RUN
SEATS AVAILABLE:300
TICKETS AVAILABLE:315
OVERBOOKING POINT:301
```

Linie (10 - 30) przyporządkowują zmiennym wartości. Linie 40 i 50 wyświetlają aktualne wartości zmiennych Y i Z. Należy zwrócić uwagę, że w linii 40 zmienna Y ma wartość 300. Linia 60 nadaje zmiennej Y nową wartość, która następnie jest wyświetlona w linii 70. Oznacza to, że zmienna Y może przyjmować różne wartości w trakcie wykonywania programu.
W linii 60 ukazany jest ponadto jeszcze jeden sposób przypisywania wartości zmiennej: można używać tej samej zmiennej (jej dotychczasowej wartości) po prawej stronie instrukcji podstawienia. Nie jest to dozwolone w algebrze, ale taki rodzaj podstawienia jest powszechnie stosowany przy pisaniu programów. Oznacza to : należy wykonać wszystkie działania używając aktualnej wartości zmiennej, zastąpić poprzednią wartość zmiennej nową, np:

```
Y = Y – 1
Y = Y + X
Y = Y / 2
Y = Y * (X+2)
```

PĘTLA FOR ... NEXT

W poprzedniej części tego rozdziału, przy omawianiu komendy GOTO, wspomniano o pętlach. Pętlą nazywa się wielokrotne wykonywanie jednej lub kilku linii programu.
Komendy FOR NEXT pozwalają stworzyć pętle, które określoną ilość razy wykonują pewne linie programu. Komenda FOR określa ilość powtórzeń wykonania pętli podając zakres wartości zmiennej. Np:

```
FOR LICZNIK = 1 TO 4
```

Komenda NEXT określa koniec pętli FOR/NEXT. Gdy program dochodzi do komendy NEXT, komputer sprawdza komendę FOR, aby stwierdzić czy zakres pętli został przekroczony. Jeżeli nie, zmienna podana w komendzie FOR (zwana zmienną sterującą) jest zwiększana o 1 i linie programu w pętli są wykonywana ponownie. Na przykład, gdyby dodać pętlę FOR/NEXT do programu podanego na początku rozdziału to działanie programu byłoby następujące:

```
10 FOR LI = 1 T0 4
20 ? "COMMODORE 64"
30 NEXT LI
RUN
COMMODORE 64
COMMODORE 64
COMMODORE 64
COMMODORE 64
```

Po dodaniu pętli FOR/NEXT nie ma już potrzeby przerywać wykonania programu klawiszem RUN/STOP.

Pętla FOR/NEXT działa następująco:

Linia 10 nadaje zmiennej LI wartość 1 i określa jednocześnie, że pętlę należy wykonywać tak długo, aż zmienna LI przekroczy wartość 4.

Linia 20 nakazuje komputerowi wyświetlić na ekranie napis COMMODORE 64.

Linia 30 mówi komputerowi, aby dodać 1 do zmiennej LI. Tak długo, jak długo zmienna LI jest w zakresie od 1 do 4 linie znajdujące się między komendami FOR i NEXT będę wykonywane ponownie (będzie ponownie wyświetlany napis COMMODORE 64). Gdy LI równa się 4 linia 20 również będzie wykonana. Potem linia 30 ponownie doda 1 do LI i komputer będzie wiedział, że zmienna LI wykroczyła poza zakres. Komputer przestanie wykonywać pętlę - program się skończy.

Aby wyraźniej ukazać działanie pętli FOR/NEXT, można dodać następne wyrażenia do komendy PRINT w linii 20.

```
10 FOR L1 = 1 T0 4
20 ? "COMMODORE 64 ";"LICZNIK=";LI
30 NEXT LI
RUN
COMMODORE 64 LICZNIK=1
COMMODORE 64 LICZNIK=2
COMMODORE 64 LICZNIK=3
COMMODORE 64 LICZNIK=4
```

Jak widać program kończy się automatycznie kiedy zmienna LI przekroczy zakres określony w komendzie FOR.

Wielkość zmiennej sterującej pętlę FOR/NEXT może być zwiększana o wartość inną niż 1. W tym celu należy dopisać na końcu komendy FOR słowo STEP i wartość o jaką ma się zwiększać zmienna. Na przykład:

```
NEW
10 FOR NB = 1 TO 10 STEP .5      Przecinek po komendzie PRINT oznacza,
20 PRINT NB,                     że kolejne liczby będą wyświetlane
30 NEXT NB                       w kolejnych 10 kolumnowych strefach
RUN
1 1.5 2 2.5
3 3.5 4 4.5
5 5.5 6 6.5
7 7.5 8 8.5
9 9.5 10

NEW
10 FOR A=2 TO 8 STEP 2
20 PRINT A
30 NEXT A

RUN
2  4  6  8
```

Zmienna sterująca pętli FOR/NEXT może być zmniejszana. W takim wypadku wartość po słowie STEP musi być ujemna. Na przykład gdyby zmienić linię 10 na:

```
10 FOR A= 8 T0 2 STEP –2
```

Działanie programu byłoby następujące:

```
RUN
8  6  4 2
```

KOMENDA IF/THEN

Komenda IF/THEN jest kolejną komendą pozwalającą na sterowanie wykonaniem programu. Komenda ta nakazuje komputerowi sprawdzić, czy podane wyrażenie jest prawdziwe. Jeżeli tak, to zostanie wykonana instrukcja znajdująca się za słowem THEN. W przeciwnym wypadku instrukcja ta zostanie pominięta i wykonana zostanie następna linia programu. Np.

```
10 X = 60
20 X = X + 1
30 IF X = 64 THEN PRINT "GOT IT" : END
40 GOTO 20
```

Można używać komendy IF do tworzenia pętli w programie lub, aby zdecydować czy dana część programu ma być wykonana czy nie. Np:

```
10 A = 0
20 IF A<= 8 THEN 40
30 END
40 ? "FRODO LIVES ";A
50 A = A + 2
60 GOTO 20
RUN
FRODO LIVES 0
FRODO LIVES 2
FRODO LIVES 4
FRODO LIVES 6
FRODO LIVES 8
```

W tym przykładzie komenda IF/THEN w linii 20 nakazuje komputerowi sprawdzić aktualną wartość zmiennej A. Jeżeli jest ona mniejsza lub równa 8 to program przechodzi od razu do wykonywania linii 40. Jeżeli A jest większe niż 8, czyli gdy wyrażenie nie jest prawdzie, komputer ignoruje polecenie znajdujące się za słowem THEN i przechodzi do linii 30.

Jeżeli wyrażenie w linii 20 jest fałszywe wykonywana jest linia 30. Linia 40 wyświetla na ekranie napis i aktualną wartość zmiennej A. Linia 50 każdorazowo zwiększa wartość A o 2. Gdy A osiągnie wartość 10, wyrażenie w linii 20 stanie się fałszywe i zostanie wykonana komenda END z linii 30 powodująca natychmiastowe zakończenie wykonywania programu.

W wyrażeniach komend IF/THEN można używać następujących operatorów relacji:

SYMBOL	ZNACZENIE
<	Mniejsze niż
>	Większe niż
=	Równe
<>	Większe lub mniejsze
>=	Większe lub równe
<=	Mniejsze lub równe

ROZDZIAŁ 5

SZERSZE INFORMACJE O JĘZYKU BASIC

- **Wstęp**
- **Prosta animacja**
- **Komenda INPUT**
- **Używanie komendy GET do wprowadzania danych**
- **Używanie komendy GET do obsługi klawiszy funkcyjnych**
- **Liczby losowe i inne funkcje**
- **Gra Zgadywanka**
- **Gra w kości**
- **Losowa grafika**

WSTĘP

Następne rozdziały przeznaczone są dla osób znających język BASIC i zasady pisania programów. Osoby uczące się dopiero języka BASIC mogę uznać niektóre informacje za zbyt skomplikowane. Jednakże niektóre przykładowe programy w dwóch rozdziałach: GRAFIKA SPRITE i TWORZENIE DŹWIĘKU zostały napisany tak, aby nawet osoby początkujące mogły je zrozumieć. Przykłady te ukazuję jak można wykorzystać wspaniałe możliwości graficzne i dźwiękowe C 64.

Aby lepiej zapoznać się z pisaniem programów w języku BASIC, należy przejrzeć literaturę wymienioną w Dodatku N niniejszej instrukcji. Dla osób zaznajomionych już z programowaniem w języku BASIC najbliższe rozdziały przynoszą wiele informacji przydatnych do zastosowania w zaawansowanych programach w języku BASIC.

PROSTA ANIMACJA

Można użyć niektórych możliwości graficznych C 64, wykorzystując wiedzę zdobytą w poprzednich rozdziałach, łącznie z kilkoma nowymi pomysłami.

Aby zobaczyć jak można wykorzystać znaki graficzne komputera trzeba wpisać zamieszczony poniżej program. Należy zwrócić uwagę, że w instrukcjach PRINT wpisane są znaki sterujące kursorem i czyszczące ekran. *Gdy w* programie znajdują się nazwy klawiszy ujęte w nawiasy na przykład (CRSR/LEWO), to w tym miejscu należy nacisnąć klawisz SHIFT i klawisz ←CRSR→. Na ekranie ukaże się graficzna interpretacja tego klawisza, tj. dwa prostokąty. Graficzną reprezentacją wciśniętych klawiszy SHIFT i CLR/HOME jest serce wyświetlone w negatywie.

```
10 REM ODBIJAJACA SIE KULKA
20 PRINT "(CLR/HOME)"
25 FOR X = 1 TO 10 : PRINT "(CRSR/DÓŁ)" ; : NEXT
30 FOR BL = 1 TO 40
40 PRINT " (SHIFT Q)(CRSR/LEWO)"
50 FOR TM = 1 TO 5
60 NEXT TM
70 NEXT BL
80 FOR BL = 1 TO 40
90 PRINT " (CRSR/LEWO)(CRSR/LEWO)(SHIFT Q)(CRSR/LEWO)"
100 FOR TM = 1 TO 5
110 NEXT TM
120 NEXT BL
130 GOTO 20
```

Uwaga: Wszystkie linie tego programu mieszczą się całkowicie w jednej linii na ekranie. Jednakże, gdyby linia programu nie mieściła się w jednej linii C 64 automatycznie przejdzie do nowej linii nawet dzieląc w środku słowo.

Po uruchomieniu tego programu na ekranie pojawia się kulka przesuwająca się z lewej strony w prawo i z powrotem. Proszę przyjrzeć się jak zostało to osiągnięte.

Linia 10 zawiera jedynie uwagi, określające co dany program ma wykonać. Wszystko co znajduje się po komendzie REM jest ignorowane przez komputer.

Linia 20 czyści ekran.

Linia 25 wysyła dziesięć poleceń przesunięcia kursora w dół. Powoduje to umieszczenie kulki na środku ekranu. Gdyby nie było tej linii, kulka przesuwałaby się na samej górze ekranu.

Linia 30 rozpoczyna pętlę przesuwania kulki o 40 pól z lewej strony w prawą.

Linia 40 wykonuje następujące czynności:

1. Wyświetla na ekranie znak spacji (odstępu), którym kasuje poprzednie położenie kulki.
2. Wyświetla na ekranie kulkę.

3. Cofa położenie kursora o jedno pole w lewo tak, aby przy następnym wykonaniu pętli znak spacji skasował obecne położenie kulki.

Linie 50 i 60 wykonują pustą pętlę, aby spowolnić przesuwanie się kulki po ekranie. Bez tych linii kulka poruszałaby się zbyt szybko by można ją było wyraźnie obserwować.
Linia 70 zamyka pętlę rozpoczynającą się w linii 30.

Przy każdym wykonaniu rozkazów w pętli kulka przesuwa się o jedno pole do przodu. Jak widać w programie tym występuje pętla wewnątrz pętli. W jednej pętli może być zawartych do dziesięciu innych pętli. Należy jedynie zwrócić uwagę, aby pętle się nie krzyżowały. Pętle muszą być ZAGNIEŻDŻONE jedna w drugiej. To znaczy, jeżeli otwiera się pętlę A, a następnie otwiera się pętlę B, to pętla B (wewnętrzna) musi kończyć się pierwsza. W ten sposób można zagnieździć do dziewięciu pętli.
Podczas pisania programu, w którym korzysta się z pętli, dobrze jest narysować strzałki od początku do końca danej pętli. Jeżeli strzałki się krzyżują, komputer nie będzie wiedział dokąd ma powrócić po napotkaniu komendy NEXT, nie będzie mógł więc wykonać programu.
Linie 80 do 120 są podobne do poprzednich tylko przesuwają kulkę z prawej strony w lewą.
Linia 130 powoduje powrót do linii 20 aby rozpocząć cały proces od nowa.

Można spróbować zmienić linię 40 na:

```
40 PRINT "(SHIFT Q)";
```

a następnie uruchomić program komendą RUN. Ponieważ nie ma znaku spacji i znaku przesunięcia kursora, kolejne kulki pozostają na ekranie tak długo, aż nie zostaną skasowane przez powracającą kulkę w drugiej części programu.

KOMENDA INPUT

Jak dotąd wszystko w programie było ustawione przed rozpoczęciem wykonywania programu instrukcją RUN. Po tej komendzie nie można było już nic zmienić czy dodać.
Komenda INPUT pozwala na wprowadzanie informacji do programu w czasie jego wykonywania. Po napotkaniu komendy INPUT program oczekuje na wpisanie odpowiedniej informacji i dopiero potem wykonuje następne instrukcje.
Aby zobaczyć jak działa instrukcja INPUT, należy wpisać NEW, nacisnąć RETURN, a następnie wpisać poniższy program:

```
10 INPUT A$
20 PRINT "WPISANO ";A$
30 PRINT
40 IF A$="STOP" THEN END
50 GOTO 10

RUN
? START                 Tu należy wpisać dowolny tekst i
WPISANO START           nacisnąć RETURN.

? DALEJ
WPISANO DALEJ

? STOP                  Wpisanie słowa STOP zakończy program
WPISANO STOP
```

Linia 10 nakazuje komputerowi wyświetlić znak zapytania i oczekiwać na wpisanie tekstu. Tekst ten przypisywany jest zmiennej A$. Program zaczyna wykonywać następne instrukcje dopiero po naciśnięciu klawisza RETURN.
Linia 20 wyświetla na ekranie komunikat oraz wpisany tekst.
Linia 30 powoduje pozostawienie dodatkowej pustej linii na ekranie.
Linia 40 zatrzymuje wykonywanie programu jeżeli wprowadzony tekst brzmiał STOP.
Linia 50 powoduje powrót do 10-tej linii i powtórzenie wszystkiego od początku. Jeżeli ostatnio wpisano słowo STOP, linia ta nie będzie już wykonywana.

Komenda INPUT pozwala na wpisanie zmiennych łańcuchowych (tekstów) lub numerycznych (liczb). Komenda ta ponadto umożliwia wypisanie komunikatu zakończonego znakiem zapytania, aby określić na jakie informacje oczekuje komputer. Na przykład zmieniając linię 10 powyższego programu otrzyma się:

```
10 INPUT "CZY DALEJ" ; A$
RUN
CZY DALEJ? DALEJ
WPISANO DALEJ
```

```
CZY DALEJ? STOP
WPISANO STOP
```

Komunikat w komendzie INPUT nie może być dłuższy niż 38 znaków.

A oto bardziej złożony przykład wykorzystujący wiele z dotychczasowych informacji, między innymi komendę INPUT.

```
NEW
1 REM PROGRAM ZAMIANY TEMPERATURY
5 PRINT "(SHIFT CLR/HOME)"
10 PRINT "ZAMIENIC ZE STOPNI FAHRENHEIT'A CZY CELSJUSZA (F/C)";
15 INPUT A$
20 IF A$ = "" THEN 10
30 IF A$ = "F" THEN 100
40 IF A$ <> "C" THEN END
50 INPUT "PODAJ STOPNIE CELSJUSZA: " ; C
60 F=(C * 9)/5 + 32
70 PRINT C ; " ST. CELSJUSZA = " ; F ; " ST. FAHRENHEIT'A"
80 PRINT
90 GOTO 10
100 INPUT "PODAJ STOPNIE FAHRENHEIT'A" ; F
110 C=(F - 32) * 5/9
120 PRINT F ; " ST. FAHRENHEIT'A =" ; C ; " ST. CELSJUSZA"
130 PRINT
140 GOTO 10
```

Linia 5 czyści ekran.

Linia 10 wyświetla na ekranie pytanie.

Linia 15 wczytuje do A$ wybraną opcję.

Linie 20 do 40 sprawdzają jaka opcja została wybrana. Linia 20 nakazuje komputerowi ponowne zapytanie o opcję, jeżeli nic nie zostało wpisane (tylko naciśnięto klawisz RETURN). Linia 30 nakazuje komputerowi przejście bezpośrednio do przeliczenia st. Fahrenheit'a na Celsjusza, jeżeli wpisano literę F. Linia 40 sprawdza czy nie zostało wpisane coś innego niż C lub F. Jeżeli tak to program zostaje zakończony. Jeżeli wpisano C program automatycznie przechodzi do linii 50 i rozpoczyna zamianę st. Celsjusza na Fahrenheita.

Może się wydawać, że niepotrzebnie tak dokładnie jest sprawdzany znak wprowadzony pierwszą instrukcją INPUT. Jest to jednak dobry zwyczaj programistów - starać się przewidzieć wszystkie możliwe przypadki. Może to później oszczędzić wielu zmartwień przy uruchamianiu większych programów.

Wracając do przykładu: gdy komputer już wie, który rodzaj przekształcenia ma wykonać, następują obliczenia, a na ekranie wyświetlane są obie wartości - podana i wyliczona. Po zakończeniu obliczeń program powraca do początku. Oto przykład obliczeń:

```
ZAMIENIĆ ZE STOPNI FAHRENHEIT'A CZY CELSJUSZA (F/C)? F
PODAJ STOPNIE FAHRENHEIT'A : ? 32
32 ST. FAHRENHEIT'A = 0 ST. CELSJUSZA
```

Po wykonaniu tego programu można zapisać go na dysk. Razem z innymi programami zawartymi w tej instrukcji może utworzyć część biblioteki programów.

UŻYWANIE KOMENDY GET DO WPROWADZANIA DANYCH

Komenda GET pozwala na wczytanie jednego znaku z klawiatury bez konieczności naciskania klawisza RETURN. W wielu przypadkach znacznie przyspiesza to wprowadzanie danych.

Gdy jest wykonywany program zawierający komendę GET, każdy naciśnięty klawisz jest przyporządkowywany zmiennej podanej w komendzie GET. Na przykład:

```
1 PRINT "(SHIFT CLR/HOME)"
10 GET A$: IF A$="" THEN 10          bez odstępu między ""
20 PRINT A$;
30 GOTO 10
```

Linia 1 czyści ekran.

Linia 10 oczekuje na wciśnięcie dowolnego klawisza.

Linia 20 wyświetla na ekranie naciśnięty klawisz.

Linia 30 powoduje powrót programu do linii 10 i oczekiwanie na następny znak. Należy pamiętać, że znaki wczytywane komendą GET nie są wyświetlane na ekranie; aby się tam pokazały trzeba użyć komendy PRINT.

Komenda IF w 10 linii jest bardzo ważna. Komenda GET, w przeciwieństwie do INPUT, nie zatrzymuje wykonywania

programu, a jedynie zwraca pusty łańcuch "", jeżeli nie naciśnięto żadnego klawisza. Tak więc, druga część linii 10 powoduje oczekiwanie na wciśnięcie dowolnego klawisza.

Można spróbować opuścić drugą część linii 10 i zobaczyć jak teraz działa program.

Aby zatrzymać wykonywanie programu należy trzymając wciśnięty klawisz RUN/STOP nacisnąć klawisz RESTORE.

Można łatwo zmienić początek programu zamiany temperatury używając komendy GET zamiast INPUT. Jeżeli program ten był zapisany na dysku, należy wczytać go komendą LOAD, po czym usunąć linię 15 i zmienić linię 20 na:

```
15                                  Usunięcie linii 15
20 GET A$: IF A$ = "" THEN 20
```

Tak zmieniony program łatwiej jest obsługiwać, ponieważ nie trzeba naciskać klawisza RETURN, a tylko klawisz C lub F aby wybrać rodzaj przeliczeń. Jeżeli tak poprawiony program ma być ponownie używany, należy jeszcze raz zapisać go komendą SAVE.

UŻYWANIE KOMENDY GET DO OBSŁUGI KLAWISZY FUNKCYJNYCH

Jak wspomniano wcześniej, klawisze z prawej strony klawiatury (f1-f8), nazywane są klawiszami funkcyjnymi i mogą być wykorzystane w programach do wielu różnych zadań.
Aby wykorzystać klawisze funkcyjne w programie należy:

1. Korzystając z komendy GET wczytać, który klawisz wciśnięto.
2. Używając komendy IF porównać klawisz naciśnięty z kodem CHR$ klawisza funkcyjnego, który ma być użyty. Każdemu klawiszowi przyporządkowany jest jeden, różny od pozostałych kod CHR$. Na przykład klawiszowi f1 odpowiada kod CHR$ 133. W Dodatku F wypisane są kody CHR$ wszystkich klawiszy.
3. Po komendzie THEN podać komendę jaka ma być wywołana danym klawiszem funkcyjnym.

Po uruchomieniu programu należy nacisnąć klawisz funkcyjny, a program wykona instrukcje znajdujące się za słowem THEN. Na przykład:

```
10 GET A$: IF A$="" THEN 10
20 IF A$=CHR$(137) THEN PRINT CHR$(14);
30 IF A$=CHR$(134) THEN PRINT "HALLO"
```

Linia 10 przypisuje zmiennej A$ wciśnięty klawisz. Tak jak w poprzednim przykładzie komenda IF powoduje oczekiwanie na wciśnięcie dowolnego klawisza.

Linia 20 sprawdza, czy został wciśnięty klawisz f2, CHR$(137). Jeżeli tak to wysyłany jest kod CHR$(14). Kod ten odpowiada jednoczesnemu naciśnięciu klawiszy C= i SHIFT, to znaczy przełącza między dużymi i małymi literami.

Linia 30 sprawdza, czy został naciśnięty klawisz f3, CHR(134). Jeżeli tak to na ekranie zostanie wyświetlony napis HALLO.

A oto kody CHR$ klawiszy funkcyjnych:

```
f1 = CHR$(133)          f2 = CHR$(137)
f3 = CHR$(134)          f4 = CHR$(138)
f5 = CHR$(135)          f6 = CHR$(139)
f7 = CHR$(136)          f8 = CHR$(140)
```

LICZBY LOSOWE I INNE FUNKCJE

C64 posiada wiele wbudowanych, gotowych funkcji, które można wykorzystywać do specjalnych zadań. Funkcje te są dostępne tak samo jak komendy języka BASIC. Przewagą takich funkcji jest to, że nie trzeba wpisywać dużej ilości komend, aby wykonać specjalistyczne obliczenia. Wystarczy jedynie wpisać nazwę funkcji, a komputer sam dokona reszty obliczeń. Do tych wbudowanych funkcji należę między innymi takie jak obliczanie pierwiastka kwadratowego (SQR), odczytanie zawartości danego adresu pamięci (PEEK), generowanie liczb losowych (RND) itd. Dodatek C zawiera spis funkcji dostępnych na komputerze C 64.

Jedną z ciekawych funkcji jest funkcja generatora liczb losowych, RND. Przy pisaniu gier, czy programów edukacyjnych, często zachodzi potrzeba skorzystania z liczby losowej. Na przykład do symulacji rzutów kostką. Oczywiście można napisać program generujący liczby losowe, ale znacznie łatwiej jest wywołać po prostu gotową funkcję RND.

Aby zobaczyć jak działa ta funkcja można wpisać poniższy krótki program:

```
NEW
10 FOR X = TO 10
```

```
20 PRINT RND(1),
30 NEXT
```

Po wpisaniu komendy RUN na ekranie pojawi się:

```
.648123935          .552833091
.123987654          .134090735
.123685436          .783465125
3.5688854E-05       .675436272
.861357246          .413154543
```

Liczby na pewno nie zgadzają się z wynikiem otrzymanym na ekranie. Byłoby to wręcz niewiarygodne gdyby się zgadzały, ponieważ program wyświetlił 10 zupełnie przypadkowych liczb.

Przy kilkukrotnym uruchomieniu programu komendą RUN widać, że wyniki za każdym razem są różne. Można jednak zauważyć pewną regułę.

Po pierwsze liczby te są zawsze z przedziału między 1 a 0, ale nigdy nie są dokładnie równe 0 ani 1. Po drugie są to liczby rzeczywiste (z liczbami po przecinku).

Teraz można spróbować napisać program symulujący rzut kostką. Jak dotąd wyniki nie są dokładnie takie, jakich by można sobie życzyć. Na początek można zmienić linię 20 i ponownie wykonać program komendą RUN.

```
20 PRINT 6 * RND(1),
RUN
3.1236472           1.52845834
2.5766563           5.65247845
2.3678596           0.56348754
4.7253443           4.87813661
0.6783245           3.64355224
```

Teraz wyniki są większe od 1, ale nadal są to liczby rzeczywiste. Aby rozwiązać ten problem należy posłużyć się następną funkcją.

Funkcja INT zmienia liczbę rzeczywistą na całkowitą. Po ponownej zmianie linii 20 program działa następująco:

```
20 PRINT INT(6 * RND(1)),

RUN
2       3       1       0
2       4       5       5
0       1
```

Tak więc jesteśmy już blisko rozwiązania, ale nadal liczby są z zakresu od 0 do 5, a nie od 1 do 6. Jako ostatni krok należy jeszcze raz zmienić linię 20:

```
20 PRINT INT(6 * RND(1)) + 1,
```

Teraz po wykonaniu programu komendą RUN uzyska się wynik, o który chodziło.

Jeżeli generowana ma być liczba rzeczywista zamiast liczby całkowitej, wzór będzie się trochę różnił, ponieważ trzeba odjąć dolną granicę zakresu od górnej. Na przykład, aby generować losowe liczby rzeczywiste z zakresu od 1 do 25 należy wpisać:

```
20 PRINT RND(1) * (25 - 1) + 1
```

Wzór ogólny na uzyskiwanie liczb losowych z pewnego zakresu wygląda następująco:

```
LICZBA = RND(1) * (GÓRNY ZAKRES - DOLNY ZAKRES) + DOLNY ZAKRES
```

GRA ZGADYWANKA

Oto przykład gry wykorzystującej liczby losowe. Gra używa nie tylko funkcji RND, ale także pokazuje jak w prosty sposób można napisać ciekawą grę.

Po wpisaniu komendy RUN komputer generuje liczbę losową LI, a zadaniem gracza jest zgadnąć tę liczbę w jak najmniejszej ilości ruchów.

```
NEW
1 REM GRA ZGADYWANKA
2 PRINT "(SHIFT CLR/HOME)"
```

```
5 INPUT "PODAJ GORNY ZAKRES LICZBY"; GZ
10 LI = INT(GZ * RND(1)) + 1
15 RU=0
20 PRINT "ZNAM JUZ LICZBE"
30 PRINT : INPUT "PODAJ LICZBE"; PR
35 RU=RU+1
40 IF PR>LI THEN PRINT"MOJA LICZBA JEST WIEKSZA": GOTO 30
50 IF PR<LI THEN PRINT"MOJA LICZBA JEST MNIEJSZA": GOT0 30
60 IF PR=LI THEN PRINT "BRAWO! ZGADLES"
65 PRINT "W ZALEDWIE"; RU ; " RUCHACH." : PRINT
70 PRINT "CZY CHCESZ ZAGRAC JESZCZE RAZ (T/N)"
80 GET OD$: IF OD$="" THEN 80
90 IF OD$ = "T" THEN 2
100 IF OD$ <> "N" THEN 70
110 END
```

Na początku programu należy podać jaki ma być górny zakres liczby do odgadnięcia, a następnie odgadnąć tę liczbę. Przykładowy przebieg gry, a potem objaśnienia.

```
PODAJ GORNY ZAKRES LICZBY? 25
ZNAM JUZ LICZBE

PODAJ LICZBE? 15
MOJA LICZBA JEST WIEKSZA

PODAJ LICZBE? 20
MOJA LICZBA JEST MNIEJSZA

PODAJ LICZBE? 19
BRAWO! ZGADLES
W ZALEDWIE 3 RUCHACH.

CZY CHCESZ ZAGRAC JESZCZE RAZ (T/N)?
```

Komendy IF/THEN (linie 40-60) porównuję liczbę podaną przez gracza z losową liczbą utworzoną w linii 10. Jeżeli gracz nie zgadł, komputer wyświetla napis informujący czy liczba była za mała, czy za duża.

Po każdej próbie odgadnięcia w linii 35 zmienna RU jest zwiększana o 1. RU jest więc licznikiem wskazującym, za którym razem gracz odgadł prawidłową liczbę. Oczywiście, celem gry jest odgadnięcie liczby w jak najmniejszej ilości ruchów.

Gdy odpowiedź jest prawidłowa program wyświetla napis BRAWO! ZGADLES, i podaje w ilu ruchach została ta liczba odgadnięta.

Przy każdej grze komputer generuje nową, losową liczbę do odgadnięcia.

Można dodać kilka linii do programu, na przykład aby program mógł także określać dolny zakres liczb generowanych w tej grze.

Uwagi dot. zapisywania programów:

W liniach 40 i 50 dwukropek oddziela kolejne komendy w jednej linii. Oszczędza się w ten sposób nie tylko czas, ale także miejsce w pamięci.

Należy zauważyć, że komendy IF/THEN w tych dwóch liniach najpierw wypisują coś na ekranie, a dopiero potem powoduję przejście do innej linii programu.

GRA W KOŚCI

Następny program symuluje rzut dwoma kostkami. Można używać tego programu jako osobnej prostej gry lub jako część większego programu.

```
5 PRINT "CHCESZ SPRAWDZIC SWOJE SZCZESCIE?"
10 PRINT "KOSTKA CZERWONA = " ; INT(RND(1) * 6) + 1
20 PRINT "KOSTKA BIALA = " ; INT(RND(1) * 6) + 1
30 PRINT "NACISNIJ KLAWISZ SPACJI BY GRAC DALEJ" : PRINT
40 GET A$:IF A$ = "" THEN 40
50 IF A$=CHR$(32) THEN 10
```

Program ten wykorzystuje informacje zawarte w poprzedniej części tej książki, więc każdy powinien sam dojść do tego jak ten program działa. CHR$(32) jest kodem oznaczającym znak spacji.

LOSOWA GRAFIKA

Na zakończenie jeszcze jeden program wykorzystujący liczby losowe, stanowiący jednocześnie wstęp do projektowania grafik. Należy wpisać i uruchomić ten program komendą RUN.

```
10 PRINT "(SHIFT CLR/HOME)"
20 PRINT CHR$(205.5 + RND(1));
30 GOTO 20
```

Funkcja CHR$ zwraca znak, który jest przypisany do danego kodu (liczby z zakresu 0 - 255). Każdy znak, który może być wyświetlony na ekranie ma przypisany kod. Dodatek F zawiera listę kodów CHR$.

Najszybszą metodą, ustalenia kodu przypisanego danemu znakowi, jest skorzystanie z funkcji ASC. Należy wpisać:

```
PRINT ASC("X")
```

X jest znakiem, którego kodu szukamy. X może być dowolnym znakiem, włączając w to znaki graficzne. Znak ten musi być ujęty w cudzysłów. Na przykład:

```
PRINT ASC("G")
71
```

Funkcja CHR$ jest funkcję przeciwną do ASC.

```
PRINT CHR$(71)
G
```

Po wpisaniu:

```
PRINT CHR$(205) ; CHR$(206)
```

na ekranie pojawią się znaki graficzne, znajdujące się po prawej stronie na klawiszach M i N, z których korzysta powyższy program.

Wyrażenie 205.5 + RND(1) przybiera wartości od 205.5 do 206.5. Jest równie prawdopodobne, że wartość ta będzie większa niż 206, jak i mniejsza niż 206. Funkcja CHR$ ignoruje część ułamkową liczby, więc w połowie przypadków będzie wyświetlany znak o kodzie 205, a w pozostałych o kodzie 206.

Można eksperymentować, zwiększając lub zmniejszając cyfrę po kropce w liczbie 205.5. Spowoduje to większe prawdopodobieństwo ukazania się jednego ze znaków.

ROZDZIAŁ 6

KOLORY I GRAFIKA

- **Jak wykorzystywać kolory i grafikę komputera**
- **Wyświetlanie kolorów**
- **Kody CHR$ kolorów**
- **Jak używać PEEK i POKE**
- **Znaki na ekranie**
- **Mapa pamięci ekranu**
- **Mapa pamięci kolorów**
- **Inny program z odbijającą się kulkę**

JAK WYKORZYSTYWAĆ KOLORY I GRAFIKĘ KOMPUTERA

Jak dotąd, książka ta przedstawiała niektóre z możliwości C 64. Jednak jedną z najciekawszych możliwości tego komputera jest korzystanie z 16 kolorów i wielu znaków graficznych.

Bardzo proste przykłady zastosowania tej grafiki przedstawione były w programie odbijającej się kulki oraz w labiryncie na końcu poprzedniego rozdziału. Ten rozdział pełniej wyjaśnia programowanie grafiki i kolorów, podaje sposoby wykorzystania ich we własnych grach i animacji.

WYŚWIETLANIE KOLORÓW

Podczas ustawiania kolorów opisanego w rozdziale 1, wyjaśniono jak zmienić kolor naciskając tylko klawisz CTRL i jeden z klawiszy kolorów.

Commodore 64 posiada 16 kolorów. Na klawiszach 1 - 8 opisanych jest ich tylko 8, zaś pozostałe kolory można uzyskać naciskając klawisz **C=** i jeden z tych klawiszy. A oto lista wszystkich kolorów:

KEYBOARD	COLOR	DISPLAY	KEYBOARD	COLOR	DISPLAY
CTRL 1	BLACK		C= 1	ORANGE	
CTRL 2	WHITE		C= 2	BROWN	
CTRL 3	RED		C= 3	LT. RED	
CTRL 4	CYAN		C= 4	GRAY 1	
CTRL 5	PURPLE		C= 5	GRAY 2	
CTRL 6	GREEN		C= 6	LT. GREEN	
CTRL 7	BLUE		C= 7	LT. BLUE	
CTRL 8	YELLOW		C= 8	GRAY 3	

W programie odbijającej się kulki, w poprzednim rozdziale pokazane było, że takie polecenia z klawiatury, jak ruch kursora mogę być wpisane do komendy PRINT. W ten sam sposób można umieszczać w programach polecenia zmiany koloru.

Aby skorzystać z tego polecenia, należy wpisać komendę NEW, a następnie spróbować wpisać poniższą linię. Po wpisaniu numeru linii, słowa PRINT i cudzysłowu nacisnąć klawisz CTRL i jednocześnie klawisz 1, zwolnić oba klawisze, nacisnąć klawisz K, potem znowu trzymając wciśnięty klawisz CTRL nacisnąć klawisz 2, zwolnić oba klawisze, nacisnąć klawisz O. I tak dalej, aż do wpisania całego słowa KOLORY.

```
10 PRINT "(CTRL 1)K(CTRL 2)O(CTRL 3)L{CTRL 4)O(CTRL 5)R(CTRL 6)Y"
```

Podobnie jak klawisze ruchu kursora miały swoje odpowiedniki graficzne, tak i w tym wypadku każde naciśnięcie klawisza CTRL i klawisza koloru powoduje pojawienie się znaku graficznego na ekranie. Z powodu tych znaków napis po komendzie PRINT może wyglądać dość dziwnie, ale po wykonaniu komendy RUN na ekranie widoczny będzie tylko napis KOLORY. Kolor kolejnych liter napisu będzie się zmieniał, odpowiednio do znaków sterujących umieszczonych w komendzie PRINT.

Można przeprowadzić własne eksperymenty, zmieniając kody kolorów w komendzie PRINT. Nie należy zapominać, że dodatkowe 8 kolorów można uzyskać naciskając klawisz C= zamiast CTRL.

Uwaga: Po uruchomieniu komendą RUN programu zmieniającego kolory napis READY i wszystko wpisane później będzie w takim kolorze, na jaki zmienił go ostatni znak sterujący w programie. Aby powrócić do normalnego zestawu kolorów należy nacisnąć jednocześnie klawisze RUN/STOP i RESTORE.

KODY CHR$ KOLORÓW

Przed przeczytaniem tej części warto zajrzeć do Dodatku F, w którym wypisane są kody CHR$ wszystkich klawiszy na klawiaturze.

Patrząc na ten spis, można zwrócić uwagę, że każdy kolor ma swój własny kod CHR$, tak jak inne klawisze. Umieszczenie kodu CHR$ w komendzie PRINT daje taki sam rezultat, jak naciśnięcie jednocześnie klawisza CTRL czy C= i odpowiedniego klawisza kolorów. Na przykład:

```
NEW
10 PRINT CHR$(147): REM (SHIFT CLR/HOME)
20 PRINT CHR$(28);"CHR$(28) ZMIENIA KOLOR NA ?"
RUN
CHR$(28) ZMIENIA KOLOR NA ?
```

Po wykonaniu komendy RUN ekran zostaje wyczyszczony, a następnie wyświetlony jest napis z linii 20. Napis ten powinien pojawić się w kolorze czerwonym.

W wielu przypadkach prościej jest używać funkcji CHR$ do zmiany kolorów, zwłaszcza gdy chce się eksperymentować. Poniżej przedstawiony jest inny program pozwalający uzyskać na ekranie różne kolory. Znaczna część linii w tym programie jest bardzo do siebie podobna (linie 40 – 110), więc korzystanie z klawiszy edycji znacznie zaoszczędzi nam czasu. Uwagi pod programem przypominają jak można to zrobić.

```
NEW
1 REM AUTOMATYCZNE PASY KOLOROWE
5 PRINT CHR${147): REM CHR$(147)=(SHIFT CLR/HOME)
10 PRINT CHR$(18);"        ";:REM 10 ZNAKOW SPACJI
20 CL = INT(8*RND(1))+1
30 ON CL GOTO 40,50,60,70,80,90,100,110
40 PRINT CHR$(5); : GOTO 10
50 PRINT CHR$(28); : GOTO 10
60 PRINT CHR$(30); : GOTO 10
70 PRINT CHR$(31); : GOTO 10
80 PRINT CHR$(144); : GOTO 10
90 PRINT CHR$(156); : GOTO 10
100 PRINT CHR$(158); : GOTO 10
110 PRINT CHR$(159); : GOTO 10
```

Linie od 1 do 40 wpisać normalnie. Na ekranie powinno to wyglądać następująco:

```
1 REM AUTOMATYCZNE PASY KOLOROWE
5 PRINT CHR$(147): REM CHR$(147)=(SHIFT CLR/HOME)
10 PRINT CHR$(18);"      ";:REM 10 ZNAKOW SPACJI
20 CL = INT(8*RND(1))+1
30 ON CL GOTO 40,50,60,70,80,90,100,110
40 PRINT CHR$(5); : GOTO 10
```

UWAGI EDYTORSKIE

Ustawić kursor na linii 40, na cyfrze 4 w liczbie 40 i nacisnąć klawisz 5. Następnie przesunąć kursor w prawo, na cyfrę 5 w nawiasie funkcji CHR$. Trzymając klawisz SHIFT nacisnąć klawisz INST/DEL, aby zrobić miejsce na drugą cyfrę i wpisać 28. Nacisnąć klawisz RETURN. Klawisz RETURN działa niezależnie od położenia kursora w danej linii.

Teraz ekran powinien wyglądać tak:

```
1 REM AUTOMATYCZNE PASY KOLOROWE
5 PRINT CHR$(147): REM CHR$(147)=(SHIFT CLR/HOME)
10 PRINT CHR$(18);"   ";:REM 5 ZNAKOW SPACJI
20 CL = INT(8*RND(1))+1
30 ON CL GOTO 40,50,60,70,80,90,100,110
50 PRINT CHR$(28); : GOTO 10
```

Nie trzeba przejmować się linią 40. Jest już ona zapamiętana w pamięci komputera. Można to sprawdzić komendą LIST. W ten sam sposób należy zmieniać numer linii i kod CHR$, aż do wpisania reszty programu. Na koniec komendą LIST sprawdzić czy wszystkie linie zostały poprawnie wpisane i uruchomić program komendą RUN.

Działanie programu powinno być jasne, poza linią 30. Krótkie wyjaśnienie co poszczególne linie oznaczają:

Linia 5 wyświetla kod CHR$(147). Jest to odpowiednik naciśnięcia klawiszy SHIFT i CLR/HOME (wyczyszczenie ekranu).

Linia 10 przełącza kolory liter i tła na odwrotne oraz wyświetla 5 znaków spacji. Najpierw widoczne są one na ekranie jako

prostokąt w kolorze jasnoniebieskim, normalnym kolorze znaków na ekranie.

Linia 20 korzysta z funkcji generatora liczb losowych aby wyznaczyć numer koloru z zakresu od 1 do 8.

Linia 30 jest odmianą komendy IF/THEN zwaną ON/GOTO, która pozwala programowi wybrać z listy numerów linię, która ma być wykonana. Jeżeli zmienna za słowem ON (w tym wypadku CL) ma wartość 1 wykonywana będzie linia pierwsza z listy (tu linia 40). Jeżeli zmienna ma wartość 2 wykonana będzie druga linia z listy, itd.

Linie 40 do 110 zmieniają kolor wyświetlania i powoduję powrót do linii 10, gdzie wyświetlany jest następny prostokąt w danym kolorze. Cały proces rozpoczyna się od nowa.

Można rozszerzyć program tak, aby wyświetlał wszystkie 16 kolorów. Należałoby dopisać kolejne numery do listy w linii 30 i następne linie programu wyświetlające kody CHR$ odpowiadające pozostałym kolorom.

JAK KORZYSTAĆ Z PEEK I POKE

Komendy PEEK i POKE pozwalają na sprawdzenie zawartości danej komórki pamięci oraz na wpisanie tam odpowiedniej wartości.

W rozdziale 4 porównywano zmienne do małych szufladek oznaczonych przez nazwę zmiennej. Należy wyobrazić sobie także pewne specjalne szufladki, znajdujące się w pewnym określonym miejscu pamięci komputera, oznaczone numerem, dalej nazywanym adresem.

W szufladkach tych komputer przechowywuje takie informacje jak: kolor tła i ramki, jakie znaki wyświetlane są na ekranie i w jakim kolorze itd.

Można zmieniać kolor tła i ramki, definiować i przesuwać po ekranie obiekty, a nawet tworzyć muzykę wpisując pod odpowiedni adres pewne wartości komendą POKE.

Należy wyobrazić sobie szufladki pamięci wyglądające tak:

53280	53281	53282	53283
2	1		

KOLOR RAMKI KOLOR TŁA

Pierwsze dwie szufladki w tym miejscu w pamięci określają kolory ramki i tła. Przykładowo podane wartości oznaczają: 2 - kolor ramki czerwony, 1 - kolor tła biały. Po wpisaniu komendy:

```
POKE 53281,7
```

i naciśnięciu klawisza RETURN kolor tła na ekranie zmieni się na żółty, ponieważ wpisano wartość 7 oznaczające kolor żółty do szufladki o adresie 53281, w której komputer przechowuje informacje o kolorze tła.

Można próbować wpisywać inne wartości do adresu 53281 i zobaczyć jakie uzyska się efekty. Poszczególnym wartościom odpowiadają następujące kolory:

0	CZARNY	8	POMARAŃCZOWY
1	BIAŁY	9	BRĄZOWY
2	CZERWONY	10	JASNOCZERWONY
3	NIEBIESKI (CYAN)	11	SZARY 1
4	PURPURA	12	SZARY 2
5	ZIELONY	13	JASNOZIELONY
6	NIEBIESKI	14	JASNONIEBIESKI
7	ŻÓŁTY	15	SZARY 3

Poniżej jest prosty program ukazujący różne kombinacje kolorów tła i ramki:

```
NEW
10 FOR BO=0 TO 15
20 FOR BA=0 TO 15
30 POKE 53280,BO
40 POKE 53281,BA
50 FOR X=1 TO 100: NEXT X
60 NEXT BA; NEXT BO
RUN
```

Aby wpisać poleceniem POKE kolejne dane kolorów ramki i tła program ten korzysta z dwóch prostych pętli. Linia 50 zawiera pustą pętlę opóźniającą.

Czasami zachodzi potrzeba sprawdzenia aktualnego koloru ramki. Należy wtedy skorzystać z funkcji PEEK.

```
? PEEK (53280) AND 15
```

Funkcja PEEK podaje cały bajt znajdujący się pod danym adresem (liczbę z zakresu 0 – 255). Ponieważ kolorów jest 16, więc używane jest tylko pół bajtu, należy dodać AND 15, aby odrzucić drogę połowę. Jeżeli powyższą komendę wpisano po wykonaniu poprzedniego programu, wyświetlona zostanie wartość 15, ponieważ jest to ostatnia wartość wpisana przez ten program pod ten adres komendę POKE.

Ogólnie funkcja PEEK pozwala na odczytanie zawartości poszczególnych "szufladek" w pamięci. Aby wyświetlać na ekranie wartości odpowiadające aktualnym kolorom ramki i tła, należy dopisać następującą linię;

```
45 PRINT CHR$(147);"RAMKA = ";PEEK(53280) AND 15,"TLO = "; PEEK(53281) AND 15
```

ZNAKI NA EKRANIE

Jak dotąd, przy wyświetlaniu znaków na ekranie komendę PRINT, komputer wyświetlał kolejno wszystkie znaki sekwencyjnie; jeden za drugim, zaczynając od pozycji kursora, chyba, że otrzymał rozkaz przejścia do następnej linii lub następnej strefy (przecinek w komendzie PRINT).

Aby wyświetlić tekst w określonym miejscu na ekranie, trzeba wysłać odpowiednią ilość znaków sterujących kursora. Zabiera to jednak dość dużo czasu.

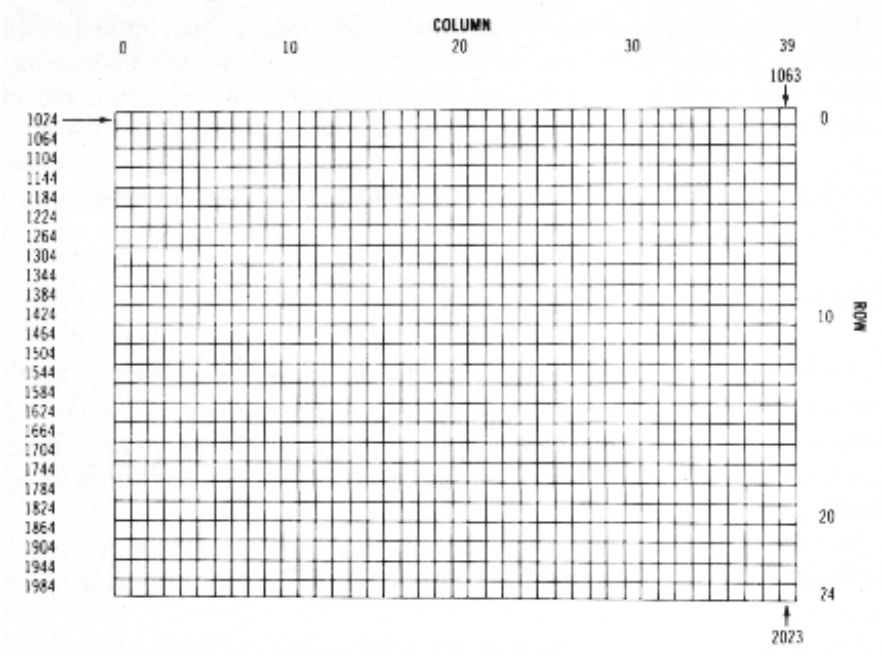
Tak jak w pewnym określonym miejscu w pamięci komputera zapisane są kolory, tak samo w innym zapisana jest zawartość ekranu. Można wykorzystywać ten fakt do bezpośredniego umieszczania znaków w danym miejscu na ekranie.

MAPA PAMIĘCI EKRANU

Na ekranie C 64 mieści się 1000 znaków (25 linii po 40 znaków), istnieje więc 1000 położonych obok siebie komórek pamięci reprezentujących kolejne znaki na ekranie. Ekran należy wyobrazić sobie jako siatkę, 40 na 25, gdzie każdy kwadrat reprezentuje jedną pozycję w pamięci.

Każda pozycja w pamięci może zawierać jeden z 256 znaków, które C 64 może wyświetlić (patrz Dodatek E). Każdy z tych znaków jest reprezentowany przez liczbę z zakresu od 0 do 255. Bezpośrednie wpisanie komendą POKE w określone miejsce w pamięci wartości odpowiadającej danemu znakowi spowoduje ukazanie się tego znaku w określonym miejscu na ekranie.

Poniżej ukazano siatkę ekranu komputera, łącznie z odpowiednimi liczbami określającymi położenie danego punktu ekranu w pamięci (adresem tego punktu).



Obszar pamięci ekranu zaczyna się od adresu (numeru) 1024 i kończy na adresie 2023. Pozycja w pamięci o adresie 1024 odpowiada punktowi na ekranie położonemu w lewym, górnym rogu. Pozycji o adresie 1025 odpowiada punkt położony o jedno miejsce w prawo itd. Pozycja o adresie 1063 określa znak w prawym górnym rogu. Następny adres określa pierwszy znak w drugiej linii. Załóżmy, że chcemy aby na środku ekranu pojawiła się kulka. Środek ekranu to pozycja w 20-tej kolumnie i 12-tej linii.

Wzór określający adres danej pozycji na ekranie wygląda następująco:

```
ADRES=1024+X+40*Y
```

gdzie X oznacza kolumnę, a Y linię.
W tym konkretnym przypadku pozycja kulki wynosi:

```
ADRES =1024 + 20 + 480
     kolumna linia(12*40)
ADRES = 1524
```

Po wyczyszczeniu ekranu klawiszami SHIFT i CLR/HOME należy wpisać:

```
POKE 1524,81
```

Po naciśnięciu klawisza RETURN, na środku ekranu pojawi się kulka (liczba 81 jest kodem oznaczającym znak kulki). Może się zdarzyć, że kulka nie będzie widoczna, jeżeli będzie wyświetlona w tym samym kolorze co tło.

MAPA PAMIĘCI KOLORÓW

Zmieniając pozycję w pamięci komputera, można zmienić kolor, w jakim kulka jest wyświetlana. Po wpisaniu:

```
POKE 55796, 2
adres, kolor
```

kulka zmieni kolor na czerwony.

Każde pole na ekranie C 64 odwołuje się do DWÓCH miejsc w pamięci komputera: jedno określa jaki znak ma być wyświetlony, a drugie w jakim kolorze. Mapa pamięci kolorów rozpoczyna się od adresu 55296 (górny lewy róg ekranu) i zajmuje następnych 1000 bajtów (pozycji). W tym obszarze zapisane są wartości określające kolor (od 0 do 15), takie same jak wartości określające kolor ramki czy tła.

Wzór na obliczenie adresy określającego kolor danej pozycji na ekranie wygląda następująco:

```
ADRES=55296+X+40*Y
```

gdzie X oznacza kolumnę, a Y linię.

INNY PROGRAM Z ODBIJAJACA SIĘ KULKA

Poniżej opisany jest program pokazujący odbijającą się kulkę, korzystający z komendy POKE i wpisujący znaki bezpośrednio do pamięci ekranu, nie korzystający z komendy PRINT. Po wpisaniu i uruchomieniu programu widać będzie, że jest on bardziej elastyczny i bardziej przybliża zagadnienia animacji.

```
NEW
10 POKE 53281,1: PRINT CHR$(5);CHR$(147)
20 POKE 53280,7; POKE 53281,6
30 X=1:Y=1
40 DX = 1: DY =1
50 POKE 1024 + X + 40*Y, 81
60 FOR X=1 TO 10 : NEXT X
70 POKE 1024 + X + 40*Y, 32
80 X=X+DX
90 IF X<=0 OR X>=39 THEN DX = - DX
100 Y=Y+DY
110 IF Y<=0 OR Y>=24 THEN DY = - DY
120 GOTO 50
```

Linia 10 ustawia kolor tła na biały i czyści ekran.

UWAGA: Czyszczenie ekranu powoduje w C 64 pracujących w systemie NTSC wypełnienie całego obszaru pamięci kolorów wartością odpowiadającą kolorowi białemu, natomiast w C 64 pracujących w systemie PAL wypełnienie aktualnym kolorem tła (w tym wypadku również kolorem białym).

Linia 20 ustala kolor ramki na żółty, a kolor tła na niebieski. Zmienne X i Y określają aktualne położenie kulki na ekranie, natomiast zmienne DX i DY określają, w którą stronę kulka ma się poruszać. Gdy do wartości X dodawane jest 1, to kulka porusza się z lewej strony w prawą, natomiast dodanie -1 do X powoduje przesuwanie się kulki w lewo. Analogicznie dodanie 1 do Y powoduje przesunięcie w dół, a -1 w górę.

Linia 50 powoduje wyświetlenie kulki w miejscu wyznaczonym przez X i Y. Linia 60 jest pętlą opóźniającą, aby kulka pozostawała na ekranie wystarczająco długo.

Linia 70 powoduje wymazanie kulki przez umieszczenie w punkcie X, Y znaku spacji (kod 32).

Linia 80 dodaje do X wartość DX określającą kierunek przesunięcia kulki.

Linia 90 sprawdza, czy kulka osiągnęła lewy lub prawy brzeg ekranu. Jeżeli tak, kierunek przesuwania się kulki zostaje zmieniony na przeciwny. Linie 100 i 110 robią to samo z Y i DY.

Linia 120 zamyka pętlę.

Można zastąpić znak kulki dowolnym innym znakiem, zmieniając wartość 81 w linii 50 na inną.

Jeżeli w linii 40 przypisze się zmiennej DX lub DY wartość 0, kulka będzie się poruszała tylko pionowo lub pozioma.

Można trochę urozmaicić ten program. Jak dotąd program sprawdzał tylko czy kulka nie osiągnęła brzegu ekranu. Można dodać następujące linie do programu:

```
21 FOR L = 1 TO 10
25 POKE 1024 + INT (RND(1)*1000), 160:
27 NEXT L
115 IF PEEK(1024 + X + 40*Y)=160 THEN DX=-DX: GOTO 80
```

Linie 21 do 27 umieszczają losowo na ekranie 10 kwadratów (kod 160 oznacza spację w odwróconych kolorach tła i ekranu). Funkcja PEEK w linii 115 sprawdza, czy w miejscu gdzie ma się pokazać kulka, nie ma kwadratu. Jeżeli jest, zmienia się kierunek ruchu kulki.

ROZDZIAŁ 7

WPROWADZENIE W GRAFIKĘ SPRITE

- **Bity i Bajty**
- **Tworzenie grafiki Sprite**
- **Projektowanie obiektu Sprite**
- **Włączanie Sprite**
- **Kolory Sprite**
- **Umieszczanie grafiki Sprite na ekranie**
- **Tworzenie kilku obiektów**
- **Priorytety**
- **Wyłączanie Sprite**

W poprzednich rozdziałach objaśniono jak wykorzystywać symbole graficzne i komendę PRINT do tworzenia animacji i innych efektów wizualnych.

W rozdziale 6 podano, jak umieszczać znaki w konkretnym miejscu na ekranie korzystając z instrukcji POKE.

W obu tych przypadkach, grafika na ekranie tworzona była z gotowych znaków graficznych, co jest pracochłonne. Aby poruszać stworzony obiekt graficzny po ekranie trzeba napisać wiele linii programu. Czasami kształt obiektu nie jest taki, jak zaplanowano, ze względu na ograniczenia wynikające z korzystania z całych znaków graficznych.

Można wyeliminować te problemy korzystając z grafiki Sprite. Sprite (dosłownie po polsku "duszek" lub "chochlik") jest to programowalny obiekt grafiki wysokiej rozdzielczości, któremu można nadać dowolny kształt korzystając z poleceń języka BASIC. Aby przesuwać taki obiekt po ekranie wystarczy podać komputerowi dokąd ten obiekt ma się przemieścić. Komputer sam zajmie się resztą.

To jeszcze nie wszystko co można robić z tymi obiektami. Można na przykład zmieniać ich kolor, badać czy dwa takie obiekty się nie zderzyły ze sobą, powodować aby jeden obiekt zasłonił drugi, a także zwiększać ich rozmiary.

Należy poznać trochę więcej szczegółów działania i posługiwania się liczbami C 64, zanim będzie można projektować swoje własne obiekty. Nie jest to jednak takie trudne, wystarczy uważnie przyjrzeć się kolejnym przykładom, a wkrótce będzie można samemu projektować obiekty w krótkim czasie.

BITY I BAJTY

Zanim rozpocznie się projektowanie grafiki Sprite, ważną rzeczą jest zrozumienie paru generalnych zasad pracy komputera.

W systemie dziesiętnym, liczy się "dziesiątkami", używając cyfr 0 - 9. Kiedy dana cyfra przekracza swoją maksymalną wartość 9, jest ona zerowana, a następna (lewa) cyfra jest zwiększana. Na przykład liczba 64 oznacza 6 * (10) + 4 * (1). Pozycja cyfry w liczbie jest istotna. Wartość 64 można obliczyć w następujący sposób:

6 * 10 ^1 + 4 * 10^0

Uwaga: Każda liczba podniesiona do potęgi 0 równa się 1.

Komputer zapamiętuje liczbę jako szereg ładunków elektrycznych, reprezentujących jedynki lub zera. Każda komórka pamięci komputera zawiera 8 takich zer lub jedynek zwanych BITAMI. Cała komórka nazywana jest BAJTEM. Pojedynczy bit, będący najmniejszą jednostką informacji może zawierać wartości 0 lub 1.

Podczas wpisywania informacji z klawiatury każde naciśnięcie klawisza zamieniane jest na kombinację 8 zer i jedynek (bajt) i wysyłane do pamięci.

Zasady arytmetyki binarnej (dwójkowej) są dużo prostsze niż innych systemów, ponieważ cyfry mogą mieć tylko dwie wartości 0 lub 1. Jak pokazano w powyższym przykładzie system dziesiętny używa jako podstawy liczby 10, podczas gdy system binarny używa liczby 2.

Jeden bit może przechowywać jedną z dwóch możliwych wartości 0 lub 1. Dwa bity mogą przechowywać jedną z 4 możliwości (4 = 2^2), natomiast trzy bity 8 (8 = 2^3). Poniżej przedstawione są zakresy liczb jedno-, dwu- i trzybitowych:

Ilość bitów	Ilość wartości	Możliwe kombinacje
1	2^1 = 2	0 1
2	2^2 = 4	00 01 10 11
3	2^3 = 8	000 001 010 011 100 101 110 111

Jak widać ilość kombinacji n bitów jest równa 2 do potęgi n-tej. Jeden BAJT mający 8 bitów może przechowywać 256 różnych wartości (256 = 2^8).

Gdy wszystkie 8 bitów ma wartość 0, bajt zawiera liczbę 0. Gdy wszystkie bity maję wartość 1, bajt zawiera liczbę 255. Bajt może zawierać wszystkie liczby z zakresu 0 - 255.

Aby zamienić wartość binarną na liczbę dziesiętną, należy dodać takie potęgi liczby 2 na jakich pozycjach bity mają wartość 1. Podany poniżej przykład pokazuje jak zapisana jest w systemie binarnym liczba 181:

POZYCJA BINARNA	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
WARTOŚĆ DZIESIĘTNA	128	64	32	16	8	4	2	1
WARTOŚĆ BITU	1	0	1	1	0	1	0	1

2^7 + 2^5 + 2^4 + 2^2 + 2^0
lub 128 + 32 + 16 + 4 + 1 = 181

Poniżej przedstawiona jest tabela ułatwiająca przeliczanie liczb binarnych na dziesiętne. Aby obliczyć wartość danego bajtu należy dodać wszystkie wartości dziesiętne, dla których bit jest równy na 1.

PRZEKSZTAŁCANIE LICZB BINARNYCH NA DZIESIĘTNE

128	64	32	16	8	4	2	1	
0	0	0	0	0	0	0	1	2^0
0	0	0	0	0	0	1	0	2^1
0	0	0	0	0	1	0	0	2^2
0	0	0	0	1	0	0	0	2^3
0	0	0	1	0	0	0	0	2^4
0	0	1	0	0	0	0	0	2^5
0	1	0	0	0	0	0	0	2^6
1	0	0	0	0	0	0	0	2^7

UWAGA: Przekształcanie liczb binarnych na dziesiętne jest podstawą projektowania obiektów sprite. Poniżej przedstawiony jest program przekształcający te liczby. Warto wpisać go i zapisać na dysku lub taśmie.

```
5 REM ZAMIANA LICZB BINARNYCH NA DZIESIETNE
10 INPUT "PODAJ 8-BITOWA LICZBE BINARNA:";A$
12 IF LEN(A$)<>8 THEN PRINT "MUSI BYC 8 BITOW!":GOTO 10
15 TL=0: C=0
20 FOR X=8 TO 1 STEP -1: C=C+1
```

```
30 TL=TL + VAL(MID$(A$,C,1)) * 2^(X-1)
40 NEXT X
50 PRINT A$;" BINARNIE =";TL;" DZIESIETNIE"
60 GOTO 10
```

W linii 10 zostaje wczytana liczba jako łańcuch A$. Linia 12 sprawdza funkcją LEN (długość łańcucha) czy wprowadzony ciąg składa się z 8 znaków. Jeżeli nie, zostaje wyświetlony komunikat i program powraca do linii 10.

W linii 15 zmienna TL oznacza wartość dziesiętną liczby binarnej, a C określa, który bit jest sprawdzany wewnątrz pętli.

Linia 30 oblicza wartość TL. Dodatek C wyjaśnia działanie funkcji VAL i MID$.

Linia 50 wyświetla liczbę binarną i jej odpowiednik w systemie dziesiętnym. Linia 60 powoduje powrót do początku programu.

TWORZENIE GRAFIKI SPRITE

Rejestry Sprite

Sprite są wyświetlane na ekranie dzięki specjalnemu układowi wewnątrz komputera nazywanemu VIC II (Video Interface Chip). Układ ten zawiera szereg bajtów nazywanych REJESTRAMI, które używane są specjalnie do obsługi obiektów Sprite.

Na przykład rejestr ENABLE SPRITE określa, który obiekt ma być pokazany na ekranie, a rejestr EXPAND określa wielkość tego obiektu. Przy wykorzystywaniu obiektów Sprite najlepiej jest traktować te rejestry jako bajty o specjalnych funkcjach. Oto lista rejestrów, używanych do obsługi Sprite.

NUMER REJESTRU OPIS

15	SPRITE POSITIONING - Pozycja obiektów Sprite
16	EXTRA MOVEMENT - Najstarsze bity pozycji
21	ENABLE - Włączanie i wyłączanie Sprite
23	EXPAND V - Rozszerzanie pionowe
27	PRIORITIES - Priorytety
28	MULTI-COLOR SELECT - Włączenie trybu kolorowego
29	EXPAND H - Rozszerzenie poziome
37-38	MULTI-COLORS - Dodatkowe kolory
39-46	COLOR - Kolory podstawowe

Każdy z rejestrów (bajtów) jest przypisany pewnemu adresowi w pamięci komputera, zaczynając od adresu 53248. Jest to tak zwany adres bazowy układu VIC II. Aby uzyskać dostęp do danego rejestru najlepiej jest przypisać pewnej zmiennej wartość adresu bazowego, a następnie dodać do niej numer rejestru. Na przykład: V=53248: POKE V+21,255 powoduje wpisanie do rejestru numer 21 wartości 255.

Układ VIC II w Commodore 64 wykonuje wszystkie czynności niezbędne, aby na ekranie ukazywały się znaki i obiekty Sprite. Należy przekazać mu jedynie następujące informacje:

Jak dany obiekt ma wyglądać.
Jakiego ma być koloru.
W którym miejscu na ekranie ma się pokazać.

Układ VIC II zawiera 46 rejestrów i może sterować do 8 obiektów Sprite. Można projektować, tworzyć i poruszać te obiekty po ekranie wpisując komendami POKE odpowiednie wartości i miejsce w pamięci.

PROJEKTOWANIE OBIEKTU SPRITE

Pojedynczy obiekt Sprite składa się maksymalnie z 504 punktów. Ułożone są one w siatkę 24 punkty szerokości na 21 punktów wysokości. Jak wspomniano, można ukazać na ekranie jednocześnie do 8 takich obiektów. Każdemu punktowi obiektu odpowiada 1 bit. Zaprojektowanie danego kształtu obiektu sprowadza się do ustawienia (nadania wartości 1) odpowiednich bitów w siatce. Każda linia siatki ma 24 punkty, co odpowiada 3 bajtom. Każdy obiekt zajmuje 64 bajty, tzn. 21 linii po 3 daje plus jeden bajt niewykorzystany (dla osób zainteresowanych: układowi łatwiej jest mnożyć razy 64, ponieważ jest to potęga liczby 2).

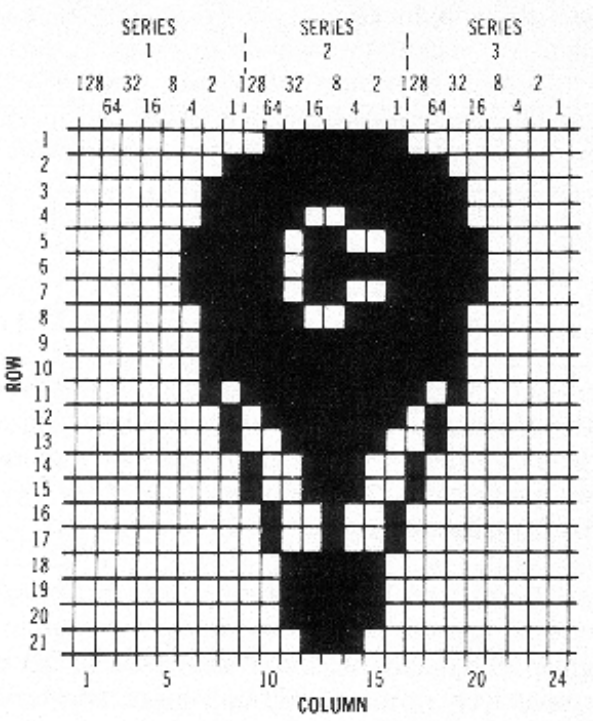
Proces projektowania jest uproszczony ponieważ można zobrazować wygląd obiektu w siatce. Załóżmy, że chcemy zaprojektować balon i sprawić by poruszał się po ekranie. Siatka na następnej stronie pokazuje jego kształt. Można też zaprojektować własny obiekt. Najlepiej jest użyć do tego kartki papieru w kratkę. Narysować siatkę o wysokości 21 linii i szerokości 24 kratek. Podzielić 24 kratki na 3 części po 8 kratek.

Następnym krokiem jest przetworzenie projektu graficznego na dane zrozumiałe dla komputera. Należy nad każdą z 8 kratek w 3 częściach wypisać kolejno liczby 128, 64, 32, 16, 8, 4, 2 i 1, są to kolejne potęgi liczby 2.

Ponumerować linie kolejnymi liczbami od 1 do 21. Teraz wypełnić siatkę dowolnym wzorem, na przykład można przerysować balon z tej instrukcji. Najprościej jest najpierw narysować zarys kształtu, a potem zamalować wybrane kratki.

Kratki zamalowane można traktować jako bity o wartości 1, natomiast puste jako 0.

Zaczynając od lewego, górnego rogu należy wszystkie kolejne sekcje 8 bitowe zamienić na liczby dziesiętne. W ten sposób uzyska się 21 linii po 3 bajty, co daje razem 63 liczby. Do zamiany liczb binarnych na dziesiętne można użyć podanego wcześniej programu.



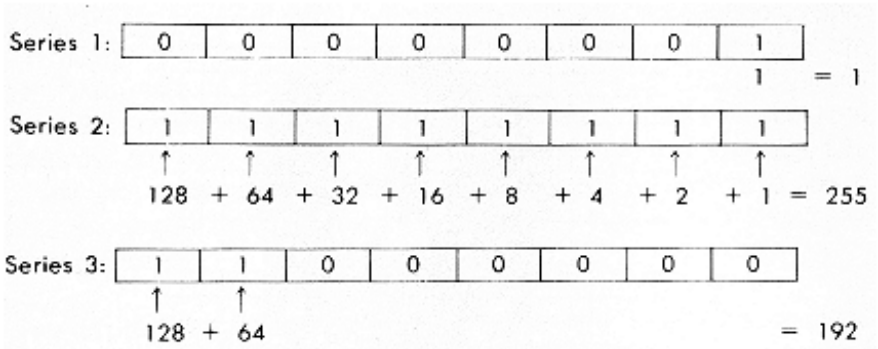
Spójrzmy teraz na projekt balonu. Pierwsza sekcja 8 punktów w pierwszej linii jest pusta, czyli wszystkie bity tego bajtu są równe 0, tzn. cały bajt równa się 0. Środkowa sekcja pierwszej linii ma następującą postać:

01111111

w zapisie dziesiętnym jest to liczba 127. Trzecia sekcja pierwszej linii jest pusta, tzn. trzeci bajt będzie równy 0. Tak więc pierwszą linię rysunku opisuję trzy liczby

0, 127, 0

Kolejne sekcje w linii 2 oblicza się w następujący sposób:



Drugą linię określają liczby

1,255,192

W ten sam sposób należy obliczyć liczby określające pozostałe linie. Po zakończeniu tych obliczeń można wpisać poniższy program w języku BASIC, jako że balon (lub dowolny inny kształt) zostały przetworzone na formę zrozumiałą przez komputer.

```
5 PRINT CHR$(147)
10 V=53248 : REM ADRES BAZOWY UKLADU VIC II
11 POKE V+21,4 : REM WLACZENIE SPRITE 2
12 POKE 2042,13 : REM DANE SPRITE 2 W BLOKU 13
20 FOR N=0 TO 62 : READ Q : POKE 832+N,Q : NEXT N
30 FOR X=0 TO 200
40 POKE V+4,X : REM WPISANIE POZYCJI X
50 POKE V+5,X : REM WPISANIE POZYCJI Y
60 NEXT X
70 GOTO 30
200 DATA 0,127,0,1,255,192,3,255,224,3,231,224
210 DATA 7,217,240,7,223,240,7,217,240,3,231,224
220 DATA 3,255,224,3,255,224,2,255,160,1,127,64
230 DATA 1,62,64,0,156,128,0,156,128,0,73,0,0,73,0
240 DATA 0,62,0,0,62,0,0,62,0,0,28,0
```

Jeżeli powyższy program został wpisany prawidłowo to, po wpisaniu komendy RUN i naciśnięciu klawisza RETURN, na ekranie pojawi się balon i będzie przesuwał się w poprzek ekranu. W chwili obecnej jeszcze nie cały program może być zrozumiały. W miarę dalszego omawiania grafiki Sprite wyjaśni się znaczenie niektórych instrukcji.

W liniach 200 - 240 wpisane są liczby opisujące wygląd obiektu. Są one kolejno odczytywane komendą READ i zapisywane w pamięci w pętli w linii 20.

WSKAŹNIKI PÓL SPRITE

Wskaźniki pól Sprite wskazują miejsce w pamięci, gdzie zapisane są liczby określające wygląd danego obiektu. Wskaźniki te znajdują się w pamięci komputera w 8 kolejnych bajtach między adresami 2040 do 2047 włącznie. Wskaźnik obiektu Sprite 0 znajduje się pod adresem 2040, obiekt 1 pod adresem 2041, itd do adresu 2047 określającego 7 obiekt.

Każdy wskaźnik może zawierać liczbę od 0 do 255. Liczba ta pomnożona przez 64 wskazuje adres, od którego wpisane są dane obiektu. Ponieważ dane jednego obiektu zajmują 64 bajty, wskaźnik może odnosić się do dowolnego bloku 64 bajtów w pierwszych 16 kB pamięci dostępnych układowi VIC II (256*64= 16384=16kB). Istnieje możliwość użycia innego bloku 16 kB pamięci.

Uwaga: Ponieważ w pamięci komputera zapisany jest program oraz inne informacje w języku BASIC, należy rozważnie wybrać adres, pod którym wpisane będą dane obiektów. Jeżeli wykorzystuje się do 3 obiektów, najlepiej jest wykorzystać bloki 13, 14 i 15 (adresy początkowe bloków danych wynoszą odpowiednio 832, 896 i 960). Jeżeli chce się korzystać z większej ilości obiektów, można korzystać z bloków od 255 (ostatniego) w dół. Jednak przy dłuższych programach, część programów może zostać zniszczona po wpisaniu danych. W takim wypadku należy zmienić blok 16kB używany przez układ VIC II.

W programie wyświetlającym obiekt na ekranie linia 10

```
V=53248
```

przypisuje zmiennej V adres bazowy układu VIC II. Później w programie adres rejestru uzyskuje się przez dodanie numeru rejestru, do tej zmiennej. Na przykład linia 11:

```
POKE V+21,4
```

przesyła dane do 21 rejestru układu.

W linii 12 programu

```
POKE 2042,13
```

wskazuje, że dane obiektu wpisane są w 13 bloku (od adresu 13 * 64 =832).

WŁĄ**CZANIE SPRITE**

Aby zobaczyć Sprite na ekranie, trzeba go uaktywnić. Służy do tego rejestr 21 układu VIC II (ENABLE). Jak wspomniano, linia 11, w programie wyświetlającym obiekt na ekranie, włącza Sprite 2. Aby to zrobić program umieszcza w rejestrze 21 wartość 4 (2^2).

Można porównać to z opisem struktury bajtu znajdującym się na początku tego rozdziału. Aby włączyć dwa obiekty Sprite jednocześnie należy dodać ich wartości. Na przykład, aby włączyć Sprite 2 i Sprite 3 należy dodać 4 (2^2) i 8 (2^3). Instrukcja będzie wtedy wyglądała następująco:

```
POKE V+21,12
```

Uwaga: Najprostszym sposobem na włączenie wybranego obiektu jest użycie prostego przeliczenia, ustawiającego dany bit w rejestrze ENABLE układu VIC II. W komendzie poniżej zmienna SN równa się numerowi obiektu, który ma być włączony (0 - 7).

```
POKE V + 21, PEEK(V + 21) OR (2^SN)
```

KOLORY OBIEKTÓW SPRITE

Sprite może mieć dowolny, jeden z 16 kolorów dostępnych na komputerze C 64. Kolory te oznaczone są numerami od 0 do 15. Rozdział 6 oraz Dodatek G zawierają listę tych kolorów i odpowiadających im numerów. Jak widać z listy rejestrów układu VIC, każdy obiekt ma swój własny rejestr koloru. Do tego celu wykorzystywane są rejestry 39 do 46. Rejestr 39 określa kolor obiektu 0, rejestr 40 określa kolor obiektu 1 itd. aż do rejestru 46 określającego kolor obiektu 7.

Gdy Sprite jest na ekranie jego punkty mają kolor określony w odpowiednim rejestrze, natomiast reszta obiektu jest przezroczysta i pokazuje kolory jakie są "pod" nim.

Aby zmienić kolor obiektu 2 na jasno zielony (kolor 13) należy wpisać instrukcję:

```
POKE V + 41, 13
```

UMIESZCZANIE OBIEKTU SPRITE NA EKRANIE

Do umieszczenia obiektu Sprite w określonym miejscu na ekranie służą następujące rejestry:

a) Rejestr pozycji X
b) Rejestr pozycji Y
c) Rejestr najstarszych bitów pozycji.

Rejestry pozycji X i Y określają miejsce, w którym na ekranie ma się pojawić Sprite. Rejestr X określa pozycję poziomą, natomiast Y pionową. Rejestry pozycji mają numery od 0 do 15, tak więc:

Rejestr 0 określa współrzędną X dla obiektu 0
Rejestr 1 określa współrzędną Y dla obiektu 0
Rejestr 2 określa współrzędną X dla obiektu 1
Rejestr 3 określa współrzędną Y dla obiektu 1

i tak dalej do rejestrów 14 i 15 określających współrzędne X i Y obiektu 7. Rejestr 16 zostanie omówiony dalej.
W celu umieszczenia obiektu w dowolnym miejscu na ekranie należy wpisać jego pozycję do odpowiedniego rejestru używając instrukcji POKE. Należy podać obie współrzędne (X i Y). Współrzędne te odnoszą się do lewego, górnego rogu obiektu, niezależnie od tego, które punkty w siatce 24 x 21 punktów są wypełnione.

Przyglądając się ponownie programowi wyświetlającemu balon na ekranie można zauważyć, że pętla FOR/NEXT w liniach 30 - 70 przesuwa obiekt ukośnie po ekranie, wpisując kolejne rosnące wartości współrzędnych X i Y do rejestrów 4 i 5 układu.

Można zauważyć, że podczas wykonywania programu balon nie przesuwa się do prawej krawędzi ekranu. Umieszczenie obiektu z prawej strony ekranu jest trudne, ponieważ ekran ma 320 punktów, a największą wartością, jaką można wpisać do rejestru jest 255. Potrzebny jest zatem jeszcze jeden bit, rozszerzający liczbę określającą położenie obiektu na ekranie. Te dodatkowe bity dla wszystkich 8 obiektów znajdują się w rejestrze 16. Bity 0 do 7 rejestru odpowiadają kolejnym obiektom. Jeżeli współrzędna X danego obiektu jest mniejsza niż 255 to odpowiadający temu obiektowi bit musi być równy 0. Natomiast aby n-ty obiekt znalazł się w miejscu o współrzędnej X1 większej od 255 należy ustawić n-ty bit rejestru 16, a do rejestru pozycji X wpisać wartość (X1 - 256). Na przykład aby przesunąć obiekt 6 z pozycji 256 do 319 należy wpisać do rejestru 16 wartość 64 (2^6):

```
POKE V+16,64
```

a następnie wpisać pętlę przesuwającą ten obiekt:

```
FOR X=0 TO 63 : POKE V + 12,X : NEXT
```

Przedstawiony niżej program przesuwa obiekt po całym ekranie:

```
10 V=53248 : POKE V+21,4 : POKE 2042,13
20 FOR N=0 TO 62 : READ Q : POKE 832+N,Q : NEXT N
25 POKE V+5,100
30 FOR X=0 TO 255
40 POKE V+4,X
50 NEXT X
60 POKE V+16,4
```

```
70 FOR X=0 TO 63
80 POKE V+4,X
90 NEXT X
100 POKE V+16,0
110 GOTO 30
200 DATA 0,127,0,1,255,192,3,255,224,3,231,224
210 DATA 7,217,240,7,223,240,7,217,240,3,231,224
220 DATA 3,255,224,3,255,224,2,255,160,1,127,64
230 DATA 1,62,64,0,156,128,0,156,128,0,73,0,0,73,0
240 DATA 0,62,0,0,62,0,0,62,0,0,28,0
```

Linia 60 ustawia najbardziej znaczący (najstarszy) bit pozycji X dla obiektu 2.
Linie 70 - 90 zawierają pętlę przesuwającą obiekt między współrzędnymi 256 i 320.
Linia 100 zeruje najbardziej znaczący bit pozycji X dla obiektu 2, tak aby obiekt pojawił się znowu w lewej części ekranu.
Gdy korzysta się z więcej niż jednego obiektu, to aby włączyć najbardziej znaczący bit dla obiektu SN, nie zmieniając tej wartości dla pozostałych, należy wpisać:

```
POKE V+16, PEEK(V+16) OR (2^SN)
```

natomiast aby go wyzerować:

```
POKE V+16, PEEK(V+16) AND (255 – 2^SN).
```

POWIĘKSZANIE OBIEKTÓW SPRITE

Obiekty można powiększać dwukrotnie w poziomie i pionie. Odpowiedzialne są za to rejestry układu VIC II.

Rejestr 23 podwaja szerokość obiektu.
Rejestr 29 podwaja wysokość obiektu.

Podobnie jak w poprzednich rejestrach, n-ty bit rejestru odpowiada n-temu obiektowi. Ustawienie danego bitu powoduje zwiększenie rozmiaru obiektu, natomiast podstawienie wartości 0 - zmniejszenie. Na przykład:

```
POKE V+23, PEEK(V+23) OR (2^SN)
```

powoduje podwojenie szerokości obiektu SN.
Użycie V+29 zamiast V+23 spowoduje podwojenie wysokości obiektu.
Dodanie linii:

```
21 POKE V+23,4:POKE V+29,4
```

do programu spowoduje, że balon poruszający się po ekranie będzie dwa razy większy.

TWORZENIE KILKU OBIEKTÓW

Tworzenie kilku obiektów jest proste. Instrukcja jak stworzyć pojedynczy obiekt została podana wcześniej. Aby dodać następny obiekt, na przykład o numerze 3, trzeba dodać lub zmienić w programie tworzącym obiekt na ekranie następujące linie:

```
11 POKE V+21,12
12 POKE 2042,13: POKE 2043,13
30 FOR X=1 TO 190
45 POKE V+6,X
55 POKE V+7,190-X
```

Linia 11 włącza obiekty 2 i 3, wpisując wartość 12 (8 + 4) do rejestru 21.
Linia 12 mówi komputerowi, że dane dotyczące Spritów 2 i 3 znajdują się w 13 bloku pamięci (od adresu 832=1364).
Linie 45 i 55 poruszają obiekt 3 po ekranie, zmieniając wartości rejestrów położenia tego obiektu.
Po uruchomieniu tego programu na ekranie pojawią się dwa balony poruszające się po ekranie. Oba obiekty są identyczne, ponieważ korzystają z tych samych danych dotyczących kształtu, zawartych w bloku 13 pamięci.
Można wprowadzić jeszcze jeden obiekt na ekran dodając następujące linie.

```
11 POKE V+21,28
12 POKE 2042,13: POKE 2043,13: POKE 2044,13
25 POKE V+23,12: POKE V+29,12
48 POKE V+8,X
58 POKE V+9,100
```

Linia 11 włącza obiekty 2,3 i 4, wpisując wartość 28 (16 + 8 + 4) do rejestru 21.

Linia 12 mówi komputerowi, że dane dotyczące wszystkich Spritów znajduję się w 13 bloku pamięci.

Linia 25 podwaja rozmiar obiektów 2 i 3 poprzez wpisanie liczby 12 (8 + 4) do rejestrów 23 i 29.

Linia 48 przesuwa obiekt 4, zmieniając jego położenie X.

Linia 58 ustala położenie Y obiektu na 100 (połowa ekranu). Dotychczasowe obiekty poruszały się w kierunkach ukośnych, ponieważ zmieniane były zarówno ich współrzędne X jak i Y. Obiekt 4 ma stałą współrzędną Y, co powoduje, że przesuwa się on tylko w kierunku poziomym.

PRIORYTETY OBIEKTÓW

Przy korzystaniu z kilku obiektów często zdarza się, że jeden z nich przesłania drugi. Istnieje ogólna zasada, że zawsze obiekt o niższym numerze będzie zasłaniał obiekty o numerach wyższych. Oznacza to, że obiekt o numerze 0 będzie zasłaniał wszystkie pozostałe obiekty, natomiast obiekt 7, będzie przez wszystkie zasłaniany. Inaczej ma się sprawa jeśli chodzi o zasłanianie normalnej zawartości ekranu przez obiekty. Istnieje specjalny rejestr o numerze 27, określający, czy dany obiekt ma zasłaniać tekst na ekranie, czy być przez niego zasłaniany. Każdy bit tego rejestru odnosi się do konkretnego obiektu. Jeżeli bit ten jest wyzerowany, to obiekt będzie zasłaniał napisy na ekranie. Jeżeli bit ten jest ustawiony, tekst będzie zasłaniał obiekt. Np.

```
POKE V+27,8
```

powoduje, że obiekt 3 będzie zasłaniany przez znaki na ekranie.

WYŁĄCZANIE OBIEKTÓW

Aby dany obiekt zniknął z ekranu należy wyzerować odpowiadający mu bit w rejestrze 21 (ENABLE). Polecenie:

```
POKE V+21, PEEK (V+28) AND (255 – 2^SN)
```

powoduje wyłączenie (zniknięcie z ekranu) obiektu o numerze SN.

OPERATORY LOGICZNE

W przykładach używane były operatory logiczne AND i OR. Operator AND był używany do zerowania danego bitu w bajcie, np. rejestru 21. Operator ten powoduje porównanie logiczne poszczególnych bitów danej liczby według zasady:

```
1 AND 1 = 1
1 AND 0 = 0
0 AND 1 = 0
0 AND 0 = 0
```

Jest to tak zwana tabela prawdy. Jak widać porównywany bit jest zerowany, o ile oba bity nie zawierają wartości 1. Zobaczmy jak wygląda działanie tego operatora w praktyce. Załóżmy, że na ekranie znajduję się dwa obiekty o numerach 2 i 3. Chcemy wyłączyć tylko obiekt 2. Zgodnie z podanym wyżej wzorem należy to wykonać poleceniem POKE V+21, PEEK(V+21) AND 251. (251=255 – 2^2). Najpierw komputer pobiera zawartość rejestru 21. Znajduje się tam wartość 12, binarnie 00001100. Liczbę 251 można przedstawić jako 11111011. Działanie można więc zapisać:

Zawartość rejestru	0	0	0	0	1	1	0	0
Liczba 251	1	1	1	1	1	0	1	1
Wynik operacji AND	0	0	0	0	1	0	0	0

Tak więc wynik operacji AND jest 00001000 czyli 8. Wpisanie tej wartości do rejestru 21 powoduje wyświetlanie tylko obiektu 3. W ten sposób obiekt 2 znika z ekranu.

Kolejnym operatorem logicznym jest OR. Tablica prawdy dla tego operatora wygląda następująco:

1 OR 1 = 1
1 OR 0 = 1
0 OR 1 = 1
0 OR 1 = 0

Gdy którykolwiek bit jest ustawiony, w wyniku bit ten też będzie ustawiony.

ROZDZIAŁ 8

DŹWIĘK I MUZYKA

- **Układ SID**
- **Przykładowy program muzyczny**
- **Granie piosenek na C 64**
- **Tworzenie efektów dźwiękowych**
- **Filtry**
- **Music Composer**

Commodore C 64 jest wyposażony w jeden z najbardziej zaawansowanych technologicznie układów tworzenia dźwięku. Rozdział ten jest tylko wprowadzeniem, przedstawiającym jak można wykorzystywać ten układ. Ogólna charakterystyka układu:

a) Regulacja głośności
b) 3 niezależne kanały
c) Tworzenie różnych przebiegów
d) Szeroki zakres częstotliwości
e) Generator obwiedni (narastanie, opadanie, podtrzymanie, wybrzmiewanie).

UKŁAD SID

Układ SID (Sound Interface Device) zawiera 29 ośmiobitowych rejestrów, numerowanych od 0 do 28, odpowiedzialnych za pewne elementy tworzenia dźwięku. W tym rozdziale omawiane będzie tylko 25 pierwszych rejestrów. Zajmują one adresy od 54272 do 54296 włącznie.

Mapa rejestrów układu SID:

REJESTR	OPIS
0- 6	VOICE 1 - Kanał 1
7-13	VOICE 2 - Kanał 2
14-20	VOICE 3 - Kanał 3
21	LOW FREQUENCY - Niska Częstotliwość
22	HIGH FREQUENCY - Wysoka Częstotliwość
23	VOLUME CONTROL AND FILTERS - Regulacja głośności i filtry.

Przed omówieniem, jak tworzony jest dźwięk, należy wpisać poniższy program. Ukazuje on tylko drobną część możliwości układu SID.

PROGRAM PRZYKŁADOWY 1

```
5 S=54272
10 FOR L=S TO S+24:POKE L,0:NEXT:REM PRZYGOTOWANIE UKLADU
20 POKE S+5,9:POKE S+6,0
30 POKE S+24,15:REM MAKSYMALNY POZIOM GLOSU
40 READ HF,LF,DR
50 IF HF<0 THEN END
60 POKE S+1,HF:POKE S,LF
70 POKE S+4,33
80 FOR T=1 TO DR:NEXT
90 POKE S+4,32:FOR T=1 TO 50:NEXT
100 GOTO 40
110 DATA 25,177,250,28,214,250
120 DATA 25,177,250,25,177,250
130 DATA 25,177,125,28,214,125
140 DATA 32,94,750,25,177,250
150 DATA 28,214,250,19,63,250
160 DATA 19,63,250,19,63,250
170 DATA 21,154,63,24,63,63
180 DATA 25,177,250,24,63,125
190 DATA 19,63,250,-1,-1,-1
```

Linia 5 nadaje zmiennej S wartość adresu bazowego układu SID. Dostęp do rejestrów układu następuje przez dodanie ich numeru do S.

Regulacja głośności

C 64 posiada 16 poziomów głośności od 0 (wyłączony) do 15 (maksymalny poziom). Do regulacji poziomu głośności służy rejestr 24 układu SID. Komendą POKE można wpisać do tego rejestru żądaną wartość. Linia 30 programu przykładowego ustawia ten poziom na maksymalny poziom (15).

Zazwyczaj poziom głośności ustawia się na początku programu. Poziom ten dotyczy wszystkich trzech kanałów.

Kanały

C 64 posiada 3 niezależne kanały, które mogę być wykorzystywane osobno lub jednocześnie. Poniżej zamieszczona jest mapa rejestrów wszystkich kanałów:

KANAŁ			
1	2	3	OPIS
0	7	14	LOW FREQ. Młodszy bajt częstotliwości
1	8	15	HIGH FREQ. Starszy bajt częstotliwości
2	9	16	LOW PULSE WIDTH Mł. bajt szerokości impulsu
3	10	17	HIGH PULSE WIDTH St. bajt szerokości impulsu
4	11	18	CONTROL REG. Rejestr sterujący
5	12	19	ATTACK/DECAY SET. Ust. narastania i opadania
6	13	20	SUSTAIN/RELEASE - podtrzymanie, wybrzmiewanie

Częstotliwość

C 64 posiada po dwa rejestry dla każdego kanału służące do określania częstotliwości (wysokości) dźwięku. Pozwala to na zapis częstotliwości dźwięku w formie 16-to bitowej. Poniżej zestawione są adresy poszczególnych rejestrów:

KANAŁ	BAJT CZĘSTOTLIWOŚCI	ADRES
1	Starszy	54273
1	Młodszy	54272
2	Starszy	54280
2	Młodszy	54279
3	Starszy	54287
3	Młodszy	54286

Aby wytworzyć dźwięk o danej wysokości należy wpisać dwie wartości do rejestrów określonego kanału. W linii 60 programu przykładowego wpisywane są wartości do rejestrów 0 i 1. W ten sposób zmieniana jest częstotliwość kanału 1.

Tworzenie dźwięku o określonej częstotliwości.

Aby uzyskać dźwięk o określonej częstotliwości należy skorzystać z następującego wzoru:

$$F=CZEST/0.06097$$

gdzie CZĘST oznacza żądaną częstotliwość.

Tak uzyskaną wartość należy teraz zamienić na dwa bajty. Na początek trzeba zmienić tę liczbę na wartość całkowitą, tzn. odrzucić część ułamkową liczby (F=INT(F)). Następnie trzeba wyznaczyć starszy bajt według wzoru:

$$HI = INT(F/256)$$

oraz bajt młodszy w/g wzoru

$$LO = F - (HI*256)$$

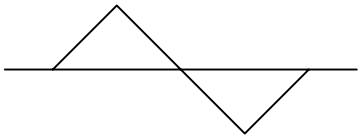
Tak uzyskane wartości należy wpisać do rejestrów częstotliwości kanału, który ma wydawać określony dźwięk.

Kształty fali

Rodzaj wybranego kształtu fali decyduje o brzmieniu czyli jakości dźwięku. C 64 może wytwarzać cztery kształty fali:

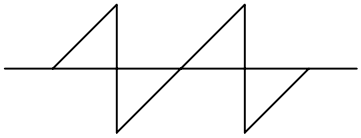
Trójkątna. Ten przebieg zawiera niewielką ilość składowych harmonicznych i wytwarza dźwięk zbliżony do dźwięku fletu. Kształt fali trójkątnej wygląda następująco:

Trójkątna:



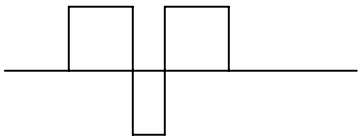
Piłokształtna: Ten przebieg zawiera wszystkie składowe harmoniczne i wytwarza pełny dźwięk. Kształt fali piłokształtnej wygląda następująco:

Piłokształtna:



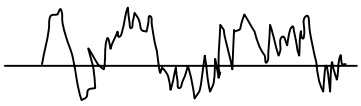
Prostokątna, o zmiennej szerokości impulsu. Zmieniając szerokość impulsu można uzyskać różne brzmienia. Tak wygląda przebieg tej fali:

Impulsy prostokątne o zmiennej szerokości:



Biały szum. Ten kształt używany jest głównie do uzyskiwania efektów dźwiękowych (np. strzały, eksplozje, szum morza itp.). Przebieg wygląda następująco:

Biały szum (głównie do efektów dźwiękowych):



Rodzaj kształtu fali określany jest w trzech rejestrach, osobno dla każdego kanału. Są to odpowiednio rejestry 4, 11 i 18. Każdy z tych rejestrów zawiera następujące informacje:

NUMER BITU OPIS

0	GATE
1 - 3	Niewykorzystane
4	Kształt trójkątny
5	Piłokształtny
6	Prostokątny
7	Szumy.

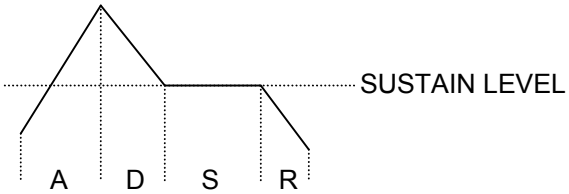
Wybranie danego kształtu następuje przez ustawienie danego bitu komendą POKE.

Bit GATE służy do sterowania generatorem obwiedni. Ustawienie go na wartość 1 powoduje rozpoczęcie cyklu Narastanie/ Opadanie/Podtrzymanie (ATTACK/DECAY/SUSTAIN). Po wyzerowaniu tego bitu rozpoczyna się cykl Wybrzmiewanie (RELEASE). Linia 70 w programie przykładowym wybiera dla kanału 1 przebieg piłokształtny i ustawia bit GATE.

Można ustawiać kombinację kilku kształtów fali, np. piłokształtną i prostokątną, jednak uzyskany dźwięk będzie bardzo dziwaczny.

Generator obwiedni

Poziom głośności pojedynczego dźwięku zmienia się od chwili rozpoczęcia emisji do momentu gdy już przebrzmi. Początkowo poziom głosu narasta od zera do pewnej wartości szczytowej. Prędkość z jaką dźwięk narasta nazywana jest Narastaniem (ATTACK). Następnie poziom dźwięku opada z wartości szczytowej do pewnej wartości pośredniej. Prędkość tego opadania określana jest jako Opadanie (DECAY). Wartość pośrednia poziomu głośności nazywana jest Podtrzymaniem (SUSTAIN). Gdy dźwięk się kończy poziom głośności opada do zera. Prędkość tego opadania nazywa się Wybrzmiewaniem (RELEASE). Poniższy rysunek przedstawia kolejne fazy dźwięku.



Uwaga: Narastanie, Opadanie i.Wybrzmiewanie są współczynnikami określającymi szybkość zmian, natomiast Podtrzymanie określa poziom głośności.

Powyższe elementy określają charakterystyczne brzmienie czyli OBWIEDNIĘ dźwięku i nazywane są parametrami dźwięku. Parametry te nazywane są w skrócie ADSR. Dla każdego kanału istnieją specjalne dwa rejestry do zapisywania parametrów ADSR. Są to rejestry 5 i 6 dla kanału 1, 12 i 13 dla kanału 2 oraz 19 i 20 dla kanału 3. Parametry Narastanie i Opadanie określane są w pierwszym z rejestrów danego kanału (5, 12, 19), natomiast Podtrzymanie i Wybrzmiewanie - w drugim (5, 13, 20).

W jednym rejestrze znajdują się po dwa parametry, ponieważ na określenie każdego z nich wystarczają tylko 4 bity, czyli pół bajtu. Pierwsze 4 bity określane są jako starsza połowa bajtu, pozostałe młodsza. Narastanie określają starsze połowy rejestru 5, 12 i 19 (zależnie od kanału), natomiast młodsze połowy tych rejestrów określają Opadanie. Podobnie starsze połowy rejestrów 6, 13 i 20 określają Podtrzymanie, a młodsze - Wybrzmiewanie.

Przed wpisaniem wartości do danego rejestru należy połączyć wartości obu parametrów, dodając odpowiednie wartości. Na przykład: Narastanie obejmuje bity o wartościach 128, 64, 32 i 16, a Opadanie bity 8, 4, 2 i 1. Załóżmy, że Narastanie ma być ustawione na 12, a Opadanie 4. Wystarczy pomnożyć Narastanie przez 16 i dodać Opadanie. W tym wypadku uzyska się wartość 12*16 + 4 =196. Wartość 196 trzeba wpisać do odpowiedniego rejestru. Zawsze do łączenia dwóch połówek bajtu (liczby 0 - 15), należy stosować wzór:

WARTOŚĆ = 16 * STARSZA POŁOWA + MŁODSZA POŁOWA.

W linii 20 programu przykładowego wartość Narastania jest ustawiana na 0, a Opadania na 9.

Maksymalna wartość zarówno Narastania, jak i opadania wynosi 15.

Kilka przykładowych instrukcji POKE:

	KANAŁ	NARASTANIE	OPADANIE
POKE 54277,66	1	ŚREDNIE(4)	MAŁE(2)
POKE 54284,100	2	ŚREDNIE(6)	ŚREDNIE(4)
POKE 54291,15	3	ZERO	MAKS.(15)
POKE 54284,255	2	MAKS.(15)	MAKS.(15)

Oto przykładowy program pokazujący jak wykorzystać rejestr Narastanie/Opadanie.

```
10 FOR L=54272 TO 54296:POKE L,O:NEXT . . . Zerowanie rej. SID
20 POKE 54296,15 . . . . . Ustaw. maks. poziom głosu
30 POKE 54277,64 . . . . . Ustaw. rej. Narast./Opad.
40 POKE 54273,162: POKE 54272,37 . . . . . Częstotliwość kanału 1
50 PRINT "NACISNIJ DOWOLNY KLAWISZ" . . .  Napis na ekranie
60 GET K$: IF K$="" THEN 60 . . . . . Sprawdzanie klawiatury
70 POKE 54276,17:FOR T=1 TO 200:NEXT . . . . Trójkątny kształt, start.
80 POKE 54276,16:FOR T=1 TO 50:NEXT . . . . . Trójkątny kształt, stop.
90 GOTO 50 . . . . . Jeszcze raz
```

Po kilkukrotnym wykonaniu programu, można zmienić obwiednię wpisując w linii 30:

```
30 POKE 54277,190
```

i ponownie uruchomić program. Zmieniając wartości Narastanie i Opadanie, można sprawdzić jak najlepiej wykorzystywać ten rejestr do tworzenia własnych efektów dźwiękowych.

Podtrzymanie i wybrzmiewanie, podobnie jak Narastanie i Opadanie, zajmują jeden rejestr. Jak już wspomniano Wybrzmiewanie jest współczynnikiem określającym szybkość opadania, natomiast Podtrzymanie określa stosunek dźwięku do wartości szczytowej, określonej w rejestrze 24. Obie te wielkości mogą mieć wartości w zakresie od 0 do 15. Należy pamiętać, że wartość Podtrzymania należy pomnożyć przez 16 i dodać wartość Wybrzmiewania.

Aby zobaczyć efekt działania tych parametrów, można dodać następującą linię do ostatniego programu przykładowego:

```
35 POKE 54278,128
```

i wykonać ten program. Linia 35 ustawia wartość Podtrzymania na 8 (około 50% maksymalnego poziomu). Można zmieniać czas trwania dźwięku, zmieniając zakres pętli w linii 70. Należy pamiętać, że wartość podtrzymania ma wpływ na poziom dźwięku, a nie na czas jego trwania.

PRZYKŁADOWY PROGRAM MUZYCZNY

Ten krótki program stanowi podsumowanie informacji zawartych w tym rozdziale.
Na początek kilka zasad dotyczących pisania programów muzycznych:

1. Wybrać kanał, który ma generować dźwięk. Jest to istotne, gdyż każdy kanał posiada swoje własne rejestry pod różnymi adresami. Można wykorzystywać dwa lub wszystkie trzy kanały jednocześnie. Ten program korzysta tylko z kanału 1.
2. Wyzerować rejestry układu SID: FOR L=54272 TO 54296: POKE L,0: NEXT
3. Ustawić poziom głosu: POKE 54296,15
4. Ustawić parametry ADSR obwiedni: POKE 54277,190:POKE 54277,248
5. Wpisać żądaną częstotliwość dźwięku do rejestrów: POKE 54273,16: POKE 54272,195
6. Rozpocząć generowanie danego kształtu fali (w tym wypadku trójkątnej): POKE 54276,17
7. Wykonać pętlę opóźniającą: FOR X=1 to 250: NEXT
8. Zakończyć dany dźwięk zerując bit GATE POKE 54276,16

A oto gotowy program:

```
NEW
5 REM SKALA MUZYCZNA
7 FOR L=54272 TO 54296 POKE L,0:NEXT . . . .  Zerowanie rej. SID
10 POKE 54296,15 . . . . . Siła głosu
20 POKE 54277,7: POKE 54278,133 . . . . . Parametry ADSR
50 READ A . . . . . Wczytanie pierwszej liczby z linii DATA
55 IF A=-1 THEN END . . . . . Zakończenie programu
60 READ B . . . . . Wczytanie następnej liczby
80 POKE 54273,A: POKE 52472,B . . . . . Ustawienie częstotliwości
85 POKE 54276,17 . . . . . Rozpoczęcie dźwięku
90 FORT=1 TO 250:NEXT:POKE54276,16 . . . . Opóźnienie i wygaszenie
95 FOR T=1 TO 50: NEXT . . . . . Czas na wybrzmienie i przerwa między nutami
100 GOTO 20 . . . . . Zamknięcie pętli
110 DATA 16,195,18,209,21,31,22,96 . . . . . Dane opisujące wysokość
120 DATA 25,30,28,49,31,165,33,135 . . . . . kolejnych nut. (Dod. M zawiera pełną listę).
999 DATA -1 . . . . . Kończy program (patrz linia 55)
```

Można zmienić kształt fali na np. piłokształtny zastępując wartości 17 i 16 w liniach 85 i 90, odpowiednio na 33 i 32. Zmiana kształtu fali zdecydowanie zmieni brzmienie dźwięku wydawanego przez komputer.

Można zmieniać dźwięk na wiele innych sposobów. Na przykład, aby zmienić brzmienie z klawesynowego na banjo trzeba zmienić parametry ADSR w linii 20 na:

```
20 POKE 54277,3: POKE 54278,0
```

Jak widać Commodore 64 może symulować brzmienie różnych instrumentów.

ODTWARZANIE PIOSENEK NA C 64

Kolejny program odtwarza kilka taktów z piosenki "Michael Row Your Boat Ashore". Program ten wykorzystuje prostokątny kształt fali, o zmiennej szerokości impulsu. Trzecia i czwarta instrukcja POKE w linii 10 definiują tę szerokość.

W pętlach opóźniających przyjęto wartość 125 odpowiadającą ósemce, 250 jako ćwierćnutę, 375 jako ćwierćnutę z kropką, 500 jako półnutę, a 1000 odpowiada długości trwania całej nuty. Przy pisaniu własnych programów można te wartości zmieniać, zależnie od własnego gustu.

```
2 FOR L=54272 TO 54296: POKE L,0: NEXT
5 S=54272
10 POKE S+24,15:POKE S+5,88:POKE S+3,15:POKE S+2,15:POKE S+6,89
20 READ H: IF H=-1 THEN END
```

```
30 READ L
40 READ D
60 POKE S+1,H: POKE S,L: POKE S+4,65
70 FOR T=1 TO D: NEXT: POKE S+4,64
80 FOR T=1 TO 50: NEXT
90 GOTO 20
100 DATA 33,135,250,42,62,250,50,60,250,42,62,125,50,60,250
110 DATA 56,99,250,50,60,500,0,0,125,42,62,250,50,60,250
120 DATA 56,99,1000,50,60,500, -1
```

Linia 2 zeruje rejestry układu SID.

Linia 5 przypisuje zmiennej S wartość adresu bazowego układu SID. Zapisanie wartości do n-tego rejestru następuje instrukcją POKE S+n,wart.

Linia 10 ustawia następujące parametry:

1. Siła głosu: POKE S+24,15
2. Kanał 1, rejestr Narast./Opad. POKE S+5,88
3. Szerokość impulsu: POKE S+3,15: POKE S+2,15
4. Kanał 1, rejestr Podtrzym./Wybrzm. POKE S+6,89

Linia 20 odczytuje liczbę z instrukcji DATA i przypisuje jej wartość zmiennej H. Jeżeli jest to wartość -1 (ostatnia w liniach DATA) program się kończy.

Linia 30 wczytuje do L kolejną liczbę z linii DATA.

Linia 40 wczytuje do D trzecią liczbę.

Linia 60 wpisuje do rejestrów następujące wartości:

1. Wartość H (pierwszą odczytaną z linii DATA) umieszczana jest w rejestrze starszego bajtu częstotliwości.
2. Wartość L do rejestru młodszego bajtu. Razem te dwie liczby określają częstotliwość danej nuty.
3. Wartość 65 do rejestru 4, rozpoczynającą odtwarzanie danej nuty.

Linia 70 wykonuje D razy pustą pętlę. W ten sposób określana jest długość trwania danego dźwięku. Jak widać liczby w liniach 100 do 120 ułożone są w trójki, z których pierwsza i druga określają starszy i młodszy bajt częstotliwości, natomiast trzecia czas trwania danej nuty. Na końcu wpisana jest wartość -1 informująca komputer, że program ma się skończyć.

Linia 80 jest pętlą opóźniająca, pozwalającą na wybrzmienie dźwięku i tworzącą krótką przerwę między kolejnymi nutami.

Linia 90 zamyka pętlę programową, odsyłając do następnej instrukcji READ.

Linie od 100 do 120 zawierają dane tworzące określoną melodię.

TWORZENIE EFEKTÓW DŹWIĘKOWYCH

Układ SID wbudowany w komputer Commodore 64 pozwala na uzyskiwanie szerokiej gamy ciekawych efektów dźwiękowych, takich jak wybuchy podczas gier, czy brzęczyk ostrzegający, że popełniono błąd podczas pracy programu.

Kilka uwag odnośnie tworzenia efektów dźwiękowych.

1. Szybka zmiana pomiędzy dwoma częstotliwościami tworzy efekt tremolo.
2. Ciekawe efekty uzyskuje się korzystając jednocześnie z kilku kanałów, niezależnie sterowanych lub gdy jednego z kanałów używa się jako echa.
3. Do różnych dźwięków używać zróżnicowanych szerokości impulsu.
4. Wykorzystywać białe szumy do tworzenia takich efektów jak strzały wybuchy, odgłosy kroków itp.
5. Wykorzystywać szeroki zakres częstotliwości i szybkie przejścia od tonów niskich do wysokich i odwrotnie.
6. Wypróbować korzystanie z dodatkowych rejestrów opisanych w Dodatku M.

Poniżej przedstawiono dwa programy ukazujące możliwości wykorzystania C 64 do tworzenia efektów dźwiękowych.

PŁACZ LALKI

```
NEW
5 FOR L=54272 TO 54296:POKE L,0:NEXT . . . .  Zerowanie rej. SID
10 S=54272: POKE 54275,15: POKE 54274,40
20 POKE S+24,15 . . . . . Poziom głosu
30 POKE S+4,65 . . . . . Włączenie przebiegu prostokątnego w kanale 1
40 POKE S+5,15 . . . . . Ustawia rejestry Narast./Opad.
50 FOR X=200 TO 5 STEP -2 . . . . . Ustawa pętlę kolejnych częstotliwości.
60 POKE S+1,40: POKE S,X: NEXT . . . . . Wpisuje kolejne częstotliwości
70 FOR X=150 TO 5 STEP -2 . . . . . Następna pętla
80 POKE S+1,40: POKE S,X: NEXT. . . . . Wpisuje kolejne częstotliwości
90 POKE S+4,0 . . . . . Wyłącza dźwięk
```

STRZELANIE

```
NEW
5 FOR L=54272 TO 54296: POKE L,0: NEXT . . . . Zerowanie rejestrów SID
10 S=54272
20 FOR X=15 TO 0 STEP –1. . . . . Pętla głośności
30 POKE S+24,X . . . . . Poziom głosu ustawiony na X
40 POKE S+4,129 . . . . . Włączenie szumów
50 POKE S+5,15 . . . . . Ustaw rejestr Narast./Opad.
60 POKE S+1,40 . . . . . Wpisuje starszy bajt częstotliowści
70 POKE S,200: NEXT . . . . . Wpisuje młodszy bajt częstotliwości
80 POKE S+5,0 . . . . . Wyłącza szumy
90 POKE S+5,0 . . . . . Wpisuje 0 do Narast./Opad.
100 GOTO 20 . . . . . Powrót do początku.
```

FILTRY

Czasami pewne kształty fal nie pozwalają uzyskać określonego brzmienia głosu. Na przykład, trudno byłoby wyobrazić takie ustawienie kształtu fali, aby dźwięk przypominał dźwięk trąbki. Aby zwiększyć możliwości odtwarzania dźwięku układ SID został wyposażony w trzy filtry.

FILTR GÓRNOPRZEPUSTOWY. Filtr ten obniża poziom wszystkich dźwięków o częstotliwości niższej od pewnej ustawionej wartości. Słyszalne są tylko wysokie dźwięki.

FILTR DOLNOPRZEPUSTOWY. Jak sugeruje nazwa filtr ten przepuszcza tylko dźwięki niskie.

FILTR PASMOWOPRZEPUSTOWY. Filtr ten przepuszcza dźwięki o częstotliwości zbliżonej do podanej. Dźwięki wysokie i niskie są obcinane.

Istnieje możliwość utworzenia FILTRU PASMOWOZAPOROWEGO przez połączenie filtrów górno- i dolnoprzepustowych. Taki filtr przepuszcza tony wysokie i niskie, obcinając środkowe pasmo.

Rejestr 24 określa, który rodzaj filtru ma być użyty. Jest to ten sam rejestr, który wykorzystywany jest do ustawiania siły głosu. Wykorzystywane są następujące bity:

NUMER BITU OPIS

4 FILTR DOLNOPRZEPUSTOWY.
5 FILTR PASMOWOPRZEPUSTOWY.
6 FILTR GÓRNOPRZEPUSTOWY.

Uaktywnienie danego filtru następuje po ustawieniu bitu odpowiadającego danemu filtrowi.
Czasami jest pożądane aby tylko część z kanałów była filtrowana. Rejestr 23 określa, których kanałów filtry dotyczą. Znaczenie poszczególnych bitów jest następujące:

NUMER BITU OPIS

7-4 Rezonans filtru (0-15)
3 Filtrować sygnał zewnętrzny
2 Filtrować kanał 3
1 Filtrować kanał 2
0 Filtrować kanał 1

Ustawienie danego bitu powoduje, że głos z danego kanału jest filtrowany.
Częstotliwość progowa filtru jest liczbą 11-bitową. Bardziej znaczące 8 bitów (11-3) zapisywane jest w rejestrze 22, natomiast pozostałe bity (2-0) w rejestrze 21. Daje to zakres wartości między 0 a 2047.
Można dodać poniższe linie do programu przykładowego i zaobserwować różnicę. W liniach tych wykorzystywany jest filtr dolnoprzepustowy.

```
30 POKE S+24,31: REM GLOSNOSC I FILTR DOLNOPRZEPUSTOWY
35 POKE S+23,1: REM WLACZENIE FILTRU DLA KANALU 1
37 POKE S+22,128: POKE 2+21,7: REM USTAWIENIE CZESTOTLIWOSCI PROGOWEJ
```

Ciekawe efekty uzyskuje się włączając i wyłączając filtry w trakcie trwania dźwięku.

MUSIC COMPOSER

Cartridge MUSIC COMPOSER pozwala na tworzenie muzyki na C 64, bez konieczności koncentrowania się na poszczególnych rejestrach układu SID. Udostępnia on wykorzystanie wszystkich możliwości układu SID poprzez wpisywanie specjalnych znaków sterujących w linie programu. Utworzony już utwór można zapisać na taśmie lub dyskietce, aby móc go ponownie odtwarzać. Podczas odtwarzania muzyki na ekranie przesuwa się pięciolinia, ukazując aktualnie grane nuty.

ROZDZIAŁ 9

ZAAWANSOWANE TECHNIKI OPEROWANIA DANYMI

- **Komendy READ i DATA**
- **Obliczanie wartości średniej**
- **Tablice**
- **Definiowanie wielkości tablic**
- **Symulacja rzutów kostką z użyciem tablic**
- **Tablice dwuwymiarowe**

KOMENDY READ I DATA

Dotychczas przedstawione były dwie metody, za pomocą których można nadawać wartość liczbową zmiennej. Przez bezpośrednie podstawienie (np. A=2) lub za pomocą komendy INPUT.

Jednakże przy przetwarzaniu większej ilości liczb obie metody mogą okazać się niepraktyczne.

W rozdziale 7 podczas przedstawiania grafiki Sprite, do określenia kształtu obiektów wykorzystywane byty komendy READ i DATA. Poniżej zamieszczony jest krótki program demonstrujący działanie tych komend.

```
10 READ X
20 PRINT "X MA WARTOSC : ";X
30 GOTO 10
40 DATA 1, 34, 10.5, 16, 234.56

RUN
X MA WARTOSC : 1
X MA WARTOSC : 34
X MA WARTOSC :10.5
X MA WARTOSC : 16
X MA WARTOSC : 234.56

?OUT OF DATA ERROR IN 10
READY
```

Komenda READ w linii 10 odczytuje wartość znajdującą się za komendą DATA i podstawia ją do zmiennej X.

Linia 30 nakazuje komputerowi powrót do linii 10, gdzie komenda READ przypisuje zmiennej X następną liczbę z linii DATA. Pętla ta powtarza się aż do momentu, gdy wszystkie liczby z linii DATA zostanę odczytane instrukcję READ.

Kilka generalnych zasad wpisywania danych w liniach DATA. W tym przypadku linia ta ma postać:

```
40 DATA 1, 34, 10.5, 16, 234.56
```

Jak łatwo zauważyć kolejne liczby są oddzielone przecinkami.

W liniach tych wolno wpisywać:

- Liczby całkowite (np. 34)
- Liczby rzeczywiste (np. 234.56)
- Liczby w zapisie naukowym (np. 2.4E+04)
- Słowa i znaki (Należy przypisywać je zmiennej łańcuchowej komendę np. READ A$),

NIE WOLNO natomiast wpisywać:

- Zmiennych (np. A)
- Wyrażeń arytmetycznych (np. 35*4)

Komenda READ odczytuje zawsze liczby znajdujące się za komendą DATA. Komendy te są ze sobą powiązane. Każda kolejna komenda READ powoduje odczytanie następnej danej za komendą DATA. W komputerze jest specjalny wskaźnik określający, która liczba została ostatnio odczytana i przesuwający się o jedną pozycję po każdej kolejnej komendzie READ. Na przykład po odczytaniu pierwszej liczby wygląda to następująco:

```
40 DATA 1, 34, 10.5, 16, 234.56
          ▲
       wskaźnik
```

Próba odczytania kolejnej liczby gdy wskaźnik przeszedł już za ostatnią liczbę, powoduje przerwanie pracy programu i wyświetlanie komunikatu OUT OF DATA ERROR.

Aby uniknąć tego błędu można zastosować pętlę FOR/NEXT:

```
NEW
10 FOR X=1 TO 3
15 READ A$
20 PRINT "A$ MA WARTOSC : ";A$
30 NEXT
40 DATA ALA, MA, ASA
RUN
A$ MA WARTOSC : ALA
A$ MA WARTOSC : MA
A$ MA WARTOSC : ASA
READY
```

W tym programie pętla FOR/NEXT określa, że komenda READ ma być wykonana tylko 3 razy, czyli tyle razy ile jest danych w linii DATA.

Czasami pisząc program nie wiemy ile danych będzie się znajdowało za komendami DATA lub po prostu nie chce się nam ich liczyć. W takich wypadkach najlepszą metodą na uniknięcie błędu OUT OF DATA jest wpisanie na końcu linii danych znacznika. Znacznikiem tym może być wartość, wyróżniająca się z pozostałych. Może to być liczba ujemna, bardzo duża, bardzo mała lub w przypadku danych łańcuchowych - specjalne słowo, na przykład STOP. Przy korzystaniu z takiego znacznika należy po komendzie READ dodać instrukcję IF/THEN kończącą pętlę jeżeli wczytany został ten znacznik. Na przykład:

```
10 READ A
15 IF A<0 THEN END
20 DATA 13, 35, 29, -999
30 PRINT "SUMA =";A
40 GOTO 10
```

Program ten odczytuje z linii DATA kolejne liczby i wyświetla je na ekranie, aż do momentu odczytania wartości ujemnej. Linia 15 nakazuje komputerowi zakończyć program po wczytaniu takiej liczby.

Istnieje możliwość ponownego użycia danych wpisanych w linii DATA. Służy do tego komenda RESTORE. Powoduje ona ustawienie wskaźnika na pierwszą liczbę, w pierwszej linii DATA programu. Można dodać do drugiego programu w tym rozdziale następujące linie:

```
50 RESTORE
60 GOTO 10
```

OBLICZANIE WARTOŚCI ŚREDNIEJ

Następny program oblicza sumę i wartość średnią liczb wpisanych za komendą DATA. Korzysta on ze znacznika określającego ilość liczb do wczytania:

```
NEW
5 T=0: CT=0
10 READ X
20 IF X=-1 THEN 50:REM SPRAWDZANIE KONCA DANYCH
25 CT=CT+1
30 T=T+X: REM OBLICZANIE SUMY
40 GOTO 10
50 PRINT "ODCZYTANO ";CT;" LICZB"
60 PRINT "SUMA =";T
70 PRINT "WARTOSC SREDNIA =";T/CT
80 DATA 75,80,62,91,87,93,78,-1

RUN
ODCZYTANO 7 LICZB
SUMA = 566
WARTOSC SREDNIA = 80.8571429
```

Linia 5 zeruje CT - licznik i T - sumę.
Linia 10 wczytuje kolejne wartości znajdujące się za komendą DATA i przypisuje je zmiennej X.
Linia 20 sprawdza czy wczytano znacznik końca danych (wartość -1). Jeżeli tak, program przechodzi do wykonywania linii 50.
Linia 25 zwiększa o 1 zawartość licznika CT.
Linia 30 dodaje kolejną wartość X do sumy T.
Linia 40 zamyka pętlę, odsyłając z powrotem do linii 10.
Linia 50, która nie jest wykonywana do momentu zakończenie pętli w liniach 10 - 40, wyświetla na ekranie zawartość licznika CT.
Linia 60 wyświetla na ekranie wartość sumy T.
Linia 70 wyświetla wartość średnią.

Komenda READ pozwala na wczytanie więcej niż jednej zmiennej jednocześnie. Mogą to być zmienne różnych typów. Np.:

```
NEW
10 READ N$,A,B,C
20 PRINT N$;" MIAL ";A;", ";B;" I ";C;" PUNKTOW"
30 PRINT "OCENA SREDNIA :";(A+B+C)/3
40 PRINT: GOTO 10
50 DATA JAN,190,185,165, ANDRZEJ,225,245,190
60 DATA ROMAN,155,185,205, LECH,160,179,187

RUN

JAN MIAL 190, 185 I 165 PUNKTOW
OCENA SREDNIA :180

ANDRZEJ MIAL 225, 245 I 190 PUNKTOW
OCENA SREDNIA : 220
```

Linia 10 odczytuje cztery wartości znajdujące się za komendą DATA i przypisuje je kolejno czterem zmiennym, w takiej kolejności, w jakiej są one wypisane.

TABLICE

Jak dotąd programy wykorzystywały tylko proste zmienne języka BASIC, takie jak X czy X$. Wprawdzie wątpliwe, aby ktoś napisał program wymagający więcej zmiennych niż wszystkie możliwe kombinacje liter i cyfr, jednakże czasem zachodzi potrzeba zgrupowania pewnych danych pod jedną nazwą. Możliwość taką daje zastosowanie tablic. Np.:

```
A(0), A(1), A(2), A(3)
```

Liczby w nawiasach nazywane są INDEKSAMI zmiennej tablicowej A. Należy zauważyć, że zmienna A1, jest pojedynczą zmienną i oznacza coś innego niż A(1).
Zaletą korzystania z tablic jest fakt, że jako indeksy mogą być użyte inne zmienne i wyrażenia. Np.:

```
A(X), A(X+1), A(4-1), A(X*2)
```

Wyrażenia w nawiasach obliczane są zgodnie z ogólnymi zasadami arytmetyki komputera.

Zmienną tablicową tworzy szereg zmiennych prostych o pewnej wspólnej nazwie. Każda z konkretnych zmiennych prostych wyróżniana jest swoim INDEKSEM. Poniżej podano prosty program, obliczający wartość średnią, wykorzystujący zmienną tablicową:

```
NEW
5 PRINT CHR$(147)
10 INPUT "PODAJ ILOSC LICZB : "; X
20 FOR A=1 TO X
30 PRINT "PODAJ LICZBE ";A;:INPUT B(A)
40 NEXT
50 SU=0
60 FOR A=1 TO X
70 SU=SU+B(A)
80 NEXT
90 PRINT:PRINT "WARTOSC SREDNIA ="; SU/X
```

```
RUN
PODAJ ILOSC LICZB : ? 5
PODAJ LICZBE 1 ? 125
PODAJ LICZBE 2 ? 167
PODAJ LICZBE 3 ? 189
PODAJ LICZBE 4 ? 167
PODAJ LICZBE 5 ? 158

WARTOSC SREDNIA = 161.2
```

Linia 5 czyści ekran.
Linia 10 pyta o ilość liczb, które będą wprowadzone komendą INPUT w linii 30.
Linia 20 rozpoczyna pętlę, w której zmienną sterującą jest A, będące jednocześnie indeksem zmiennej tablicowej B.
Linia 30 pyta o kolejne liczby i podstawia ich wartości do tablicy B z kolejnym indeksem A.
Linie 50 do 80 obliczają sumę wprowadzonych liczb.
Linia 90 wyświetla wartość średnią wprowadzonych liczb.
Przy każdym kolejnym wykonaniu pętli wartość A jest zwiększana o 1, więc kolejne wartości są wpisywane na kolejne miejsca w tablicy B. Po zakończeniu programu tablica B ma postać:

B (1)	125
B (2)	167
B (3)	189
B (4)	167
B (5)	158

Po wprowadzeniu komendą INPUT kolejne wartości przechowywane są w tablicy B. Można je odczytać podając indeks danej wartości. Na przykład można dopisać na końcu programu następujące linie:

```
100 PRINT B(X-1)
110 PRINT B(3)
120 PRINT B(X-3)
```

DEFINIOWANIE WIELKOŚCI TABLIC

Próba wpisania więcej niż dziesięciu elementów do tablicy spowoduje wyświetlenie komunikatu BAD SUBSCRIPT ERROR. Tablice, które mają zawierać więcej niż 10 elementów muszę zostać wcześniej zadeklarowane komendę DIM. Na przykład, aby używać tablicy o 25 elementach, należy wpisać w programie następująca komendę:

```
DIM B(25)
```

Do deklaracji wielkości tablicy można również używać zmiennej. Na przykład aby wpisać dowolne ilość liczb w poprzednim programie należy dopisać linię:

```
15 DIM B(X)
```

Należy zwrócić uwagę na fakt, że wielkość tablicy można zdefiniować tylko raz w trakcie wykonywania programu.
Gdy w programie używanych jest kilka tablic, można zdefiniować je w jednej linii programu. Na przykład:

```
10 DIM A(12), B$(35), C(3,5)
```

Tablica A jest zwykłą, omówioną już wcześniej tablicą, B$ jest tablicą łańcuchów tzn. elementy B$(1), B$(2) itd. są ciągami znaków. Tablica C jest zmienną dwuwymiarową. Taki rodzaj tablic będzie opisany dalej.

SYMULACJA RZUTÓW KOSTKA Z UŻYCIEM TABLIC

Przy pisaniu bardziej złożonych programów, okazuje się, że zastosowanie zmiennych tablicowych znacznie skraca ilość linii niezbędnych do napisania programu.
Na przykład, pojedyncza tablica może przechowywać informację, ile razy wypadła dana ilość oczek przy rzucie kostką:

```
NEW
1 REM SYMULACJA RZUTÓW KOSTKA
5 PRINT CHR$(147)
7 DIM OC(6)
10 INPUT "ILE RZUTOW ";X
20 FOR L=1 TO X
30 R=INT (RND(1)*6)+1
40 OC(R)=OC(R)+1
50 NEXT
60 PRINT:PRINT "OCZEK","LICZBA RZUTOW"
70 FOR C=1 TO 6: PRINT C,OC(C): NEXT
```

Linia 10 pyta ile razy ma być symulowany rzut kostką.
Linia 20 rozpoczyna pętlę, która będzie wykonywana X razy.
Linia 30 przypisuje zmiennej R wynik kolejnego rzutu.
Linia 40 określa ile razy wypadła dana liczba oczek. Na przykład, gdy R=2 to element OC(2) jest zwiększany o 1.
Linia 70 wyświetla na ekranie informację ile razy wypadły kolejne ilości oczek.
Przykładowy przebieg programu:

```
ILE RZUTOW ? 1000
OCZEK LICZBA RZUTÓW
      1       148
      2       176
      3       178
      4       166
      5       163
      6       169
```

Można porównać ten program z analogicznym, nie wykorzystującym jednak zmiennej tablicowej:

```
5 PRINT CHR$(147)
10 INPUT "ILE RZUTOW ";X
20 FOR L=1 TO X
30 R=INT (RND(1)*6)+1
40 IF R=1 THEN O1=O1+1:NEXT
41 IF R=2 TREN O2=O2+1:NEXT
42 IF R=3 THEN O3=O3+1:NEXT
43 IF R=4 THEN O4=O4+1:NEXT
44 IF R=5 THEN O5=O5+1:NEXT
45 IF R=6 THEN O6=O6+1:NEXT
60 PRINT:PRINT "OCZEK","LICZBA RZUTOW"
70 PRINT 1,O1
71 PRINT 2,O2
72 PRINT 3,O3
73 PRINT 4,O4
74 PRINT 5,O5
75 PRINT 6,O6
```

Jak widać program jest znacznie dłuższy niż w przypadku użycia tablicy. Spróbujmy sobie wyobrazić jak wyglądałby ten program gdyby na kostce można było wyrzucić 100 różnych ilości oczek !

TABLICE DWUWYMIAROWE

Jak już wcześniej wspomniano można zadeklarować tablice dwuwymiarowe. Na przykład tablicę B(4,6) można sobie wyobrazić jako tabelę o 4 rzędach i 6 kolumnach, gdzie każde pole tej tabeli może zawierać jedną wartość. Zawsze jako pierwsza podawana jest ilość rzędów. W praktyce deklaracja B(4,6) tworzy tablicę o rozmiarach 5 rzędów na 7 kolumn, ponieważ C 64 jako pierwszy indeks przyjmuje wartość 0 a nie 1. Można to przedstawić następująco:

	0	1	2	3	4	5	6
0							
1							
2							
3							
4							

W każdym z pól można wpisać dowolną wartość odwołując się do niego przez podanie dwóch indeksów. Np.:

A(3,4)=255

powoduje wpisanie do 3 rzędu 4 kolumny wartości 255.

	0	1	2	3	4	5	6
0							
1							
2							
3					255		
4							

Przy korzystaniu z tablic dwuwymiarowych obowiązują te same zasady co przy tablicach jednowymiarowych:

Deklaracja: DIM A(20,20)
Przypisanie danych tablicy: A(1,1)=374
Przypisanie danych innej zm.: AB=A(1,1)
Wyświetlanie na ekranie: PRINT A(1,1)
itd.

Poniżej przedstawiony jest przykładowy program wykorzystujący tablicę dwuwymiarową. Zestawienie danych z kwestionariusza zawierającego 4 pytania i 3 możliwe odpowiedzi na każde z tych pytań (TAK, NIE, NIE WIEM).
Tablica użyta w programie ma więc postać:

	TAK	NIE	NIE WIEM
PYTANIE 1			
PYTANIE 2			
PYTANIE 3			
PYTANIE 4			

Linie 30-65 wyświetlają kolejny numer pytania i oczekują na wprowadzenie odpowiedzi.
Linia 70 zwiększa o 1 odpowiedni element tablicy. Należy pamiętać, że R oznacza rząd, a K kolumnę.
Linia 90 pyta czy będzie wprowadzana większa ilość danych.
Linie 130-170 wyświetlają łączne zestawienie danych.

```
NEW
10 DIM A(4,3)
20 PRINT CHR$(147)
30 FOR R=1 TO 4
40 PRINT "PYTANIE ";R
50 PRINT "1 - TAK, 2 - NIE, 3 - NIE WIEM"
60 PRINT "JAKA JEST ODPOWIEDZ :";
61 GET K: IF K<1 OR K>3 THEN 61
65 PRINT K:PRINT
70 A(R,K)=A(R,K)+1
80 NEXT R
85 PRINT
90 PRINT ("NASTEPNE DANE (T/N) ?")
100 GET A$: IF A$="" THEN 100
110 IF A$="T" THEN 30
120 IF A$<>"N" THEN 100
130 PRINT CHR$(147);" ZESTAWIENIE":PRINT
140 PRINT SPC(18);"ODPOWIEDZI"
141 PRINT "PYTANIE","TAK","NIE","NIE WIEM"
142 PRINT "-------------   ---------------------------------"
150 FOR R=1 TO 4
160 PRINT R, A(R,1), A(R,2), A(R,3)
170 NEXT R
RUN
```

PYTANIE 1
1 - TAK, 2 - NIE, 3 - NIE WIEM
JAKA JEST ODPOWIEDZ : 1

PYTANIE 2
1 - TAK, 2 - NIE, 3 - NIE WIEM
JAKA JEST ODPOWIEDZ : 3

i tak dalej ...

ZESTAWIENIE

	ODPOWIEDZI		
PYTANIE	TAK	NIE	NIE WIEM
1	6	1	0
2	5	2	0
3	7	0	0
4	2	4	1

DODATKI

WSTĘP

Może nie wszyscy z Państwa wiedzą, że firma COMMODORE działa na rynku komputerowym już ponad 25 lat. W latach siedemdziesiątych COMMODORE wyprodukowało pierwszy komputer osobisty (PET). Własne laboratoria pozwalają Firmie zaprojektować i wytwarzać własne układy scalone, co pozwala na tworzenie nowych komputerów osobistych w cenie niższej, niż należałoby się spodziewać z uwagi na ich poziom techniczny.

Commodore zwraca także szczególną uwagę na odpowiednią jakość obsługi klientów. Przeprowadza szkolenie sprzedawców, popiera magazyny publikujące artykuły opisujące nowe produkty tej firmy oraz producentów oprogramowania. Zachęcamy wszystkich do spotkań w klubach użytkowników, gdzie można dzielić się swoimi pomysłami i doświadczeniami. Firma wydaje dwa magazyny zawierające programy i informacje przydatne użytkownikom komputerów COMMODORE (patrz Dodatek N).

W USA stworzono specjalną sieć komputerową, do której mogą mieć dostęp wszyscy użytkownicy COMMODORE i taniego modemu VICMODEM. Dzięki tej sieci można kontaktować się z innymi użytkownikami w celu wymiany doświadczeń.

Dodatki zawierają zestawienia, tabele i inne informacje, które mogą ułatwić szybsze i efektywniejsze programowanie C 64. Zawierają one ponadto informacje na temat urządzeń peryferyjnych, które mogą być podłączone do C 64 oraz literaturę przydatną do lepszego poznania komputera.

DODATEK A

ROZSZERZANIE SYSTEMU KOMPUTEROWEGO

Komputer C 64 posiada dużą moc obliczeniową, pozwalającą na zastosowanie go do edycji tekstów czy bazy danych.

Podstawowy zestaw składa się z komputera, odpowiedniego telewizora lub monitora i magnetofonu, na którym można przechowywać programy. Dla niektórych programów praca z magnetofonem jest zbyt wolna. Można ominąć to ograniczenie korzystając ze stacji dysków w celu zapisywania i wczytywania programów.

Dlaczego warto używać stacji dysków ?

Jeżeli programy i zbiory danych są bardzo małe, przechowywanie programów na taśmach nie powoduje specjalnych kłopotów. Jednakże z czasem może okazać się, że efektywne wykorzystanie komputera utrudnione jest przez długi czas potrzebny na wczytywanie i zapisywanie na taśmę. Jest to najwyższy czas aby zastanowić się nad zakupem stacji dysków.

STACJA DYSKÓW VIC 1541

Stacja dysków VIC 1541 pozwala na zapisanie do 144 programów lub zbiorów danych na pojedynczej dyskietce 5,25 cala. Przy korzystaniu z dyskietek, nie trzeba przejmować się takimi rzeczami jak przewinięcie taśmy do początku, czy niebezpieczeństwo skasowania poprzedniego programu w trakcie zapisywania nowego. Stacja 1541 jest "urządzeniem inteligentnym", tzn. wykonuje wszystkie czynności nie zajmując pamięci komputera. Na każdej dyskietce można pomieścić ponad 174000 znaków, jest to objętość przeciętnej powieści Dickensa. Główną przewagą stacji dysków nad magnetofonem jest jej prędkość. Operacje zapisu i odczytu są dokonywane do 40 razy szybciej, niż przy użyciu magnetofonu. Stacja VIC 1541 nie wymaga żadnego dodatkowego interface, dołączana jest bezpośrednio do komputera.

Jeśli zachodzi taka potrzeba, można dołączyć do komputera do pięciu stacji dysków. Można wtedy wczytywać i zapisywać na nich dane bez konieczności zmiany dyskietki.

DRUKARKA IGŁOWA VIC 1525

Zastosowanie drukarki znacznie zwiększa możliwości wykorzystania systemu komputerowego. Każdy system komputerowy powinien być w nią wyposażony. Pozwala ona na wydruk zestawień, listów lub programu komputerowego w celu łatwiejszej analizy. VIC 1525 jest drukarką igłową. Oznacza to, że każdy znak tworzony jest kolejnymi uderzeniami igieł. Dzięki temu każdy znak dostępny z klawiatury komputera, jak również samodzielnie zaprojektowany, może zostać wydrukowany na papierze. Drukarka VIC 1525 drukuje znaki o gęstości 12 znaków/cal i z szybkością 30 znaków/sekundę. Można używać papieru perforowanego lub pojedynczych arkuszy zwykłego papieru o szerokości do 10 cali. Drukarkę tę można podłączyć bezpośrednio do komputera.

DWUKIERUNKOWA DRUKARKA VIC 1526

Drukarka VIC 1526 różni się od VIC 1525 dwoma rzeczami. Po pierwsze ma możliwość drukowania podczas obu przesuwów głowicy drukującej. Oznacza to, że drukuje zarówno od strony prawej do lewej, jak i od lewej do prawej. Dzięki temu drukarka nie traci czasu na cofnięcie głowicy po każdym wydrukowanym wierszu.

Drugą różnicą między 1526 a 1525 jest szybkość drukowania. Czas potrzebny na wydrukowanie linii jest bezpośrednio uzależniony od ilości znaków w tym wierszu. Przy wydruku 80 kolumn prędkość wynosi 45 linii/min.; przy 40 kolumnach - 78

linii/min.; przy 20 kolumnach - 124 linie/min. Drukarka 1526 pozwala na drukowanie do 3 egzemplarzy jednocześnie (oryginał i 2 kopie). Posiada łatwą do wymiany kasetę z taśmą barwiącą. Można używać papieru o szerokości do 10 cali. Przyłącza się ją bezpośrednio do komputera.

PRINTER/PLOTTER VIC 1520

VIC 1520 może być wykorzystywany zarówno jako zwyczajna drukarka, jak i jako plotter, czyli urządzenie pozwalające zaprojektować i wykonać w czterech kolorach na papierze wykresy słupkowe, liniowe i kołowe, a także dowolny zaprojektowany kształt. Można drukować jednocześnie litery małe i duże, a także znaki graficzne. Dodatkowo istnieje możliwość definiowania wielkości pisma. Prędkość pisania wynosi około 14 znaków/sek. Zależnie od wielkości pisma, można drukować od 10 do 40 znaków w linii. Litery mogą być nawet pisane w pionie, po obróceniu o 90 stopni. Dowolne linie rysowane są przez podanie współrzędnych początkowych i końcowych oraz kształtu linii. Można ponadto zmieniać grubość linii od linii ciągłych do przerywanych. Dokładność plottera wynosi 0.2 mm, a prędkość rysowania 60 mm/sek. Plotter używa małych pisaków kulkowych, wybieranych przez użytkownika w trakcie rysowania. VIC 1520 może być podłączony bezpośrednio do komputera.

MONITOR KOLOROWY 1701

Tak uniwersalny komputer jak COMMODORE 64, powinien być podłączony do odpowiedniego monitora abyśmy mogli w pełni docenić jego możliwości. Specjalnie w tym celu został zaprojektowany kolorowy monitor 1701. Monitor ten ma przekątną 14 cali i doskonałą rozdzielczość. Dźwięk może być odtwarzany przez wewnętrzny głośnik lub w prosty sposób można podłączyć go do zestawu Hi-Fi.

JOYSTICK 1311

Joystick może być używany nie tylko do sterowania grami, lecz także, wraz z odpowiednim oprogramowaniem, jako narzędzie do tworzenia rysunków.

OPROGRAMOWANIE

Szeroka gama programów dostępnych na COMMODORE 64 obejmuje programy przydatne w domu, pracy, rozrywce oraz pomoce dla programistów.

Programy do pracy i dla domu

EASY FILE EFI 6440 (dyskietka)

EASY FILE jest kompletnym systemem bazy danych dla COMMODORE 64. Zawiera wszystkie elementy dostępne w drogich programach, za jedynie część ich ceny. Użytkownik sam określa formę w jakiej informacje się ukazują i czy mają być one drukowane na drukarce, czy wyświetlane na ekranie. Oznacza to, że EASY FILE może być stosowany zarówno w pracy, jak i w domu.

CLUB MANAGER CMG 6440 (dyskietka)

Program CLUB MANAGER został zaprojektowany jako pomoc ułatwiająca zarządzanie klubami sportowymi, stowarzyszeniami i organizacjami, w których prowadzona jest kartoteka członków. Program ten pozwala na przechowywanie informacji o wszystkich członkach na dyskietce, zamiast w kartotekach. Dane te mogą być poprawiane, gdy zachodzi taka potrzeba, zapisywane, dodawane lub kasowane z akt członkowskich. CLUB MANAGER pozwala na wypisanie listy członków, ich adresów, a także, ponieważ jest on powiązany z edytorem tekstów EASY SCRIPT można pisać standardowe listy adresowane do każdego z członków. CLUB MANAGER pozwala ponadto na tworzenie list rezerwacji, w które można wpisać kiedy dany członek ma zamiar używać kortów tenisowych, stołu bilardowego czy innego elementu wyposażenia klubu. Dzięki temu nie istnieje groźba rezerwacji jednego urządzenia przez dwie osoby w tym samym czasie. CLUB MANAGER pozwala także na stworzenie rozkładu dnia, w którym można wpisywać przewidywane zajęcia dla danych osób. Program ten jest przeznaczony dla właścicieli i sekretariatów klubów. Znacznie skróci on czas potrzebny na obsługę kartoteki członków i listy rezerwacji.

FUTURE FINANCE FFI 6440 (dyskietka)

Program FUTURE FINANCE jest programem pozwalającym na tworzenie planów finansowych. Pozwala on przewidzieć dochód firmy i obrót gotówki w oparciu o założoną wielkość produkcji i sprzedaży oraz koszty. Poszczególne elementy można zmieniać, aby zobaczyć wpływ tych zmian na całość działalności firmy. Można uzyskać zestawienie ukazujące wpływ danego produktu na zysk i przepływ gotówki w określonym czasie.

EASY CALC ECL 6440 (dyskietka)

EASY CALC jest arkuszem kalkulacyjnym. Zawiera wszystkie elementy typowe dla tego typu programów - definiowaną wielkość strony, przenoszenie informacji z jednego miejsca w inne itp. Ponadto program EASY CALC posiada wiele dodatkowych możliwości takich jak: biblioteka funkcji trygonometrycznych, statystycznych i innych, możliwość rysowania wykresów w dowolnym miejscu na arkuszu, ochrona dostępu do danych przez użycie hasła i wiele wiele innych.

EASY STOCK EST 6440 (dyskietka)

Program EASY STOCK jest programem ułatwiającym prowadzenie magazynu. Dane dotyczące wszystkich produktów w magazynie wpisywane są z klawiatury, a następnie przechowywane na dysku. EASY STOCK pozwala na zmiany cen zarówno poszczególnych pozycji, jak i całych grup artykułów. Dostęp do informacji o danej pozycji uzyskuje się poprzez podanie numeru tej pozycji. Wszystkie dane mogą być uaktualniane, dodawane lub usuwane ze stanu magazynu. Jeżeli ilość danego produktu w magazynie spadnie poniżej pewnej określonej wartości, nazwa tego produktu wypisywana jest innym kolorem niż pozostałe. EASY STOCK pozwala na tworzenie szerokiej gamy raportów i zestawień, takich jak aktualny stan magazynu, analiza przepływu towarów i wiele innych.

EASY SCRIPT ESC 6440 (dyskietka)

Program EASY SCRIPT jest profesjonalnym edytorem tekstów. Pozwala on na szybkie tworzenie, poprawianie i wydruk tekstów. EASY SCRIPT może być używany do pisania listów, raportów, notatek jednym słowem wszystkich rodzajów tekstów. Tekst może być przechowywany na dyskietkach lub taśmie magnetofonowej, tak więc można go potem zmieniać i drukować według potrzeb.

EASY SPELL ESP 6440 (dyskietka)

EASY SPELL jest programem sprawdzającym prawidłową pisownię słów w tekstach napisanych przy pomocy edytora EASY SCRIPT. Pozwala on na kontrolę pisowni zarówno pojedynczego zbioru tekstu, jak i grupy tekstów powiązanych ze sobą. Wraz z programem sprzedawana jest dyskietka zawierająca słownik wyrazów, z którymi porównywane są kolejne wyrazy tekstu. Na razie dostępny jest tylko słownik wyrazów angielskich.

Pomoce dla programistów

Firma Commodore udostępnia wiele programów użytkowych, mających za zadanie ułatwienie pisania programów w językach BASIC i Assemblerze (kodzie maszynowym).

SIMONS' BASIC SIB 6410 (cartridge)

SIMONS' BASIC został zaprojektowany aby ułatwić programistom wykorzystanie wszystkich potencjalnych możliwości komputera COMMODORE 64. Cartridge SIMONS' BASIC to właściwie trzy programy w jednym. Zawiera on narzędzia ułatwiające wpisywanie programów, grupę rozkazów pozwalających na korzystanie z grafiki i dźwięku w C 64 oraz komendy pozwalające na programowanie strukturalne w języku BASIC, dzięki czemu programy są bardziej czytelne i łatwiejsze w pisaniu. Programy te są sprzedawane w postaci kasety cartridge, co powoduje, że ich instalacja polega na wsunięciu kasety do gniazda z tyłu komputera i włączeniu komputera. Wszystkie nowe komendy SIMONS' BASIC wpisuje się tak samo jak pozostałe komendy języka BASIC.

Komendy narzędziowe zawierają:

- AUTO - Automatyczne numerowanie linii.
- RENUMBER - Automatyczne przenumerowanie linii programu,
- KEY - Przypisanie komend klawiszom funkcyjnym.

i wiele innych.

Komendy graficzne zawierają:

- HIRES - Włączenie trybu graficznego wysokiej rozdzielczości.
- REC - Rysowanie prostokąta.
- CIRCLE - Rysowanie okręgu.
- PAINT - Zamalowywanie wnętrza dowolnej figury.

oraz komendy do tworzenia i obsługi obiektów Sprite oraz znaków graficznych definiowanych przez użytkownika:

- DESIGN - Włączenie siatki do projektowania kształtu obiektu lub wyglądu znaku.
- MMOB - Poruszanie obiektów Sprite po ekranie.

- DETECT - Wykrywanie kolizji obiektów.

i wiele, wiele innych.

Komendy programowania strukturalnego zawarte w SIMONS' BASIC stanowię pomoc dla programistów o każdym poziomie umiejętności. Pozwalają one na nadanie pewnym fragmentom programu nazw i wywoływanie tych fragmentów przez nazwę, a nie numer linii, który łatwo można zapomnieć. Ponadto udostępnione są inne komendy ułatwiające programowanie:

- PROC - Nadanie nazwy fragmentowi programu.
- CALL - Skok do nazwanego fragmentu.
- EXEC - Wywołanie nazwanego fragmentu programu i powrót do następnej komendy po zakończeniu wykonywania tego fragmentu.
- REPEAT..UNTIL - Pętla powtarzająca się aż zostanie spełniony określony warunek.

oraz wiele innych.

Ponadto SIMONS' BASIC zawiera komendy pozwalające na formatowanie wyświetlanych tekstów na ekranie, przesuwanie zawartości ekranu, kontrolę danych wprowadzanych komendą INPUT, operacje na łańcuchach znaków, zamianę liczb szesnastkowych, dziesiętnych i binarnych, dzielenie liczb całkowitych i wiele, wiele innych. Dodatkowo SIMONS' BASIC zawiera grupę rozkazów, pozwalających na obsługę błędów. Można wyświetlać własne komunikaty o błędach. Zakres komend dostępnych dzięki SIMONS' BASIC czyni z niego praktyczne narzędzie dla każdego programisty. Współpracuje ze stacją dysków i magnetofonem.

ASSEMBLER TUTOR AST 6440 (dyskietka) AST 6420 (taśma)

Program ASSEMBLER TUTOR jest bardzo przydany dla tych wszystkich, którzy chcą się nauczyć programowania w kodzie maszynowym. Jest on także bardzo wartościowy dla tych programistów, którzy już posiadają pewną wiedzę na temat programowania w assemblerze, jednak chcieliby dowiedzieć się więcej o kodzie maszynowym mikroprocesora 6502. ASSEMBLER TUTOR podzielony jest na trzy moduły. Każdy z tych modułów obejmuje jeden z aspektów assemblera i zawiera wstęp, krótki test oraz omówienie różnych zagadnień dotyczących programowania w assemblerze.

ASSEMBLER DEVELOPMENT ASM 5440 (dyskietka)

Program ASSEMBLER DEVELOPMENT pozwala na pisanie programów dla COMMODORE 64 bezpośrednio w kodzie maszynowym. Zawiera wszystkie narzędzia niezbędne do tworzenia, wczytywania z dysku i wykonywania programów w kodzie mikroprocesora 6510.

PROGRAMMER'S UTILITIES UTL 6440 (dyskietka)

PROGRAMMER'S UTILITIES zawiera wiele procedur (zarówno dla zaawansowanych programistów, jak i dla początkujących) pozwalających na maksymalne wykorzystanie COMMODORE 64. Znajduję się tam między innymi procedury obsługi dysku, jak procedura zmieniająca numer stacji dysków czy kopiująca zawartość dyskietki przy jednej stacji dysków, procedury graficzne takie jak projektowanie obiektów Sprite i własnych znaków graficznych, komendy obsługujące dźwięk i formatujące wyświetlanie tekstów na ekranie.

Rozrywka

Firma Commodore produkuje nie tylko szeroką gamę wspaniałych gier zręcznościowych, ale także wiele programów strategicznych dla tych, którzy wolą raczej trenować swój umysł niż palec naciskający przycisk Fire w joysticku. W takich grach ograniczeniem nie jest czas, a zdolności poradzenia sobie w złożonej sytuacji.

LABIRYNTH LBY 6420 (taśma)

Czy odważysz się wejść do labiryntu? Czy uda Ci się z niego wydostać, czy pozostaniesz tam na zawsze? Możesz przez chwilę spojrzeć gdzie jesteś, ale zabiera Ci to punkty. Jest to gra w której możesz się zagubić.

HIGH FLYER HFL 6440 (dyskietka)

Jest rok 1945. Wojna właśnie się skończyła i postanowiłeś założyć własne linie lotnicze. Dzięki odpowiedniemu zarządzaniu udało Ci się prowadzić przedsiębiorstwo od 1945 roku aż do dzisiaj. Wszystkie decyzje należą do Ciebie: rozkład lotów, wybór najlepszej proporcji w przewozie pasażerów i ładunków aby uzyskać maksymalny dochód; kiedy jest najlepszy czas na zakup nowych samolotów; czy wzięć kredyt z banku, a jeśli tak to z którego itd. Czy potrafisz zachować pozycję, którą udało Ci się osiągnąć? Dostępne tylko na dyskietkach.

RAIL BOSS RBO 6440 (dyskietka)

Jesteś pionierem budowy kolei i masz za zadanie wybudować linię kolejową na amerykańskim Dzikim Zachodzie między Base City a Junction City. Twoim zadaniem jest nająć robotników i strażników chroniących Ciebie i budowę przed włóczącymi się po okolicy bandytami. Budując stacje po drodze powodujesz przypływ pieniędzy pozwalających Ci na opłacenie swoich pracowników i zakup niezbędnych materiałów. Twoje zamiary mogą pokrzyżować wojowniczy Indianie, którzy próbują przerwać pracę zabijając twoich pracowników i niszcząc linię kolejową. Twoją jedyną nadzieją jest, że oddział kawalerii z Fort Commodore zdąży złapać Indian, zanim oni złapią Ciebie. Jest to program dla przyszłych pionierów z całego świata.

OCEAN RACER OCR 6440 (dyskietka)

Bierzesz udział w regatach jachtem dookoła świata. Regaty zaczynają się w Portsmouth. Trasa wiedzie przez Cape Town w Afryce Południowej, Auckland w Nowej Zelandii, Rio de Janeiro w Brazylii i z powrotem do Portsmouth. Możesz wybrać jacht, na którym będziesz kapitanem: jednomasztowy slup, dwumasztowy kecz lub jednomasztowy katamaran. Regaty odbywają się w czterech etapach, z których każdy zawiera wiele niebezpieczeństw od gór lodowych do uszkodzenia jachtu w wyniku zderzenia z przepływającym wielorybem. Musisz zdecydować jaki kurs wybierzesz i jakie żagle postawisz przy danym wietrze. Gra dla starych wilków morskich i ludzi z aspiracjami na żeglarzy.

Jest to tylko pierwsza seria gier umysłowych zaprojektowana specjalnie na COMMODORE 64. Cały czas opracowywane są nowe gry. Szczegóły na ten temat można znaleźć w magazynach komputerowych i w klubach.

DODATEK B

OPIS KOMUNIKATÓW BŁĘDÓW DOS

UWAGA: Komunikaty o błędach DOS o numerach mniejszych niż 20 można zignorować, w wyjątkiem błędu 01, który informuje ile zbiorów zostało skasowanych komendą SCRATCH.

20 READ ERROR (nagłówek bloku nie znaleziony)
Kontroler dysku nie może znaleźć wskazanego bloku danych. Może to być spowodowane podaniem niewłaściwego numeru bloku lub fizycznym uszkodzeniem dyskietki.

21 READ ERROR (brak sygnału synchronizującego)
Kontroler dysku nie może wykryć sygnału synchronizującego na danej ścieżce. Może to być spowodowane nieprawidłowym położeniem głowicy, brakiem dyskietki, użyciem niesformatowanej dyskietki lub nieprawidłowym jej wsunięciem. Może też oznaczać nieprawidłowe funkcjonowanie stacji dysków.

22 READ ERROR (brak bloku danych)
Kontroler dysku dostał polecenie odczytu lub weryfikacji bloku danych, który nie został prawidłowo zapisany. Błąd ten występuje w połączeniu z komendami blokowymi i oznacza żądanie dostępu do nieistniejącego sektora i/lub ścieżki.

23 READ ERROR (błąd sumy kontrolnej w bloku danych)
Ten komunikat wskazuje, że nastąpił błąd przy odczycie jednego lub więcej bloków danych. Dane zostały wczytane do pamięci stacji lecz nastąpił błąd sumy kontrolnej na końcu bloku. Może to wskazywać na niewłaściwe uziemienie.

24 READ ERROR (błąd dekodowania bajtu)
Dane lub nagłówek zostały wczytane do pamięci stacji lecz wystąpił błąd sprzętowy, ponieważ stwierdzono nieprawidłową kombinację bitów w bajcie danych. Może to wskazywać na niewłaściwe uziemienie.

25 WRITE ERROR (błąd zapis-weryfikacja)
Błąd ten występuje gdy kontroler stacji stwierdzi różnicę między danymi zapisanymi, a danymi w pamięci stacji.

26 WRITE PROTECTION ON
Ten komunikat ukazuje się podczas próby zapisu na dyskietkę zabezpieczoną (z zalepionym wycięciem z boku dyskietki).

27 READ ERROR (błąd sumy kontrolnej w nagłówku bloku)
Kontroler stwierdził błąd sumy kontrolnej w nagłówku bloku, z którego miał odczytywać dane. Blok nie został wczytany do pamięci stacji. Może to wskazywać na niewłaściwe uziemienie.

28 WRITE ERROR (długi blok danych)
Kontroler próbuje wykryć sygnał synchronizujący następnego nagłówka po zapisaniu bloku danych. Jeżeli nie wykryje tego sygnału w ciągu pewnego czasu wyświetlany jest powyższy komunikat o błędzie. Błąd ten spowodowany jest nieprawidłowym formatem dyskietki lub błędem sprzętowym.

29 DISK ID MISMATCH
Ten komunikat jest wyświetlany gdy kontroler dostał polecenie dostępu do niesformatowanej dyskietki lub gdy jej nagłówek jest uszkodzony.

30 SYNTAX ERROR (ogólna składnia)
DOS nie potrafi zinterpretować polecenia wysłanego kanałem poleceń. Zazwyczaj jest to spowodowane nieprawidłową ilością nazw zbiorów, na przykład dwie nazwy podane po lewej stronie komendy COPY.

31 SYNTAX ERROR (nieprawidłowe polecenie)
DOS nie rozpoznaje polecenia. Polecenie musi zaczynać się od pierwszej pozycji.

32 SYNTAX ERROR (nieprawidłowe polecenie)
Przesłano polecenie dłuższe niż 58 znaków.

33 SYNTAX ERROR (nieprawidłowa nazwa zbioru)
Nazwa zbioru została nieprawidłowo podana w komendzie OPEN lub SAVE.

34 SYNTAX ERROR (brak nazwy zbioru)
W poleceniu dla DOS brakuje nazwy zbioru lub DOS jej nie rozpoznał jako takiej. Zazwyczaj brak dwukropka (:) w poleceniu.

39 SYNTAX ERROR (nieprawidłowe polecenie)

Ten błąd występuje gdy polecenie przesłane kanałem poleceń (kanał 15) jest nierozpoznane przez DOS.

50 RECORD NOT PRESENT

Ten błąd jest wynikiem próby czytania komendami INPUT# lub GET# więcej znaków niż zawiera zbiór. Komunikat ten pojawia się również po ustawieniu wskaźnika poza końcem zbioru o dostępie swobodnym (REL). Jeżeli zamierza się powiększyć ten zbiór przez dodanie nowego rekordu (komendą PRINT#), błąd ten należy zignorować. Komendy GET i INPUT nie powinny być ponawiane przed zmianą pozycji wskaźnika zbioru.

51 OVERFLOW IN RECORD

Komenda PRINT# wysłała więcej znaków niż objętość rekordu. Nadmiar znaków zostaje zignorowany. Należy pamiętać, że kod klawisza RETURN oznaczający koniec rekordu doliczany jest do długości rekordu.

52 FILE TOO LARGE

Pozycja wskaźnika w zbiorze o dostępie swobodnym (REL) wskazuje na pole nie mieszczące się na dysku. Dysk pełny.

60 WRITE FILE OPEN

Błąd ten występuje przy próbie otwarcia do czytania zbioru, który był przedtem otwarty do zapisu i nie został zamknięty.

61 FILE NOT OPEN

Błąd ten występuje przy próbie dostępu do zbioru, który nie był otwarty przez DOS. Czasami w takim przypadku błąd ten nie jest generowany, a polecenia są ignorowane.

62 FILE NOT FOUND

Żądanie dostępu do zbioru nie istniejącego na danej dyskietce.

63 FILE EXISTS

Nazwa zbioru tworzonego istnieje już na dyskietce.

64 FILE TYPE MISMATCH

Typ zbioru otwieranego różni się od typu zapisanego w katalogu dysku.

65 NO BLOCK

Ten komunikat pojawia się podczas stosowania komendy B-A. Wskazuje on, że blok który ma być oznaczony jako zajęty, był już zajęty. Parametry po komunikacie wskazują na najbliższy wolny blok na dysku. Jeżeli parametry te wynoszą 0 oznacza to, że wszystkie bloki o numerach wyższych są zajęte.

66 ILLEGAL TRACK AND SECTOR

DOS próbował odczytać ścieżkę lub blok nie istniejący na danej dyskietce. Może to wynikać z błędu przy odczycie wskaźnika następnego bloku.

67 ILLEGAL SYSTEM T OR S

Specjalny komunikat wskazujący na niewłaściwą ścieżkę lub sektor systemowy.

70 NO CHANEL (dostępny)

Żądanie dostępu do zajętego kanału lub wszystkie kanały transmisji są zajęte. Można jednocześnie otworzyć do pięciu zbiorów sekwencyjnych. Poprzez kanały bezpośredniego dostępu można otworzyć sześć zbiorów.

71 DIRECTORY ERROR

Wskazania BAM nie pokrywają się z obliczeniami. BAM w pamięci stacji mógł zostać zapisany innymi danymi. Aby poprawić ten błąd należy ponownie zainicjować stację, co spowoduje ponowne przepisanie BAM do pamięci. Niektóre z otwartych zbiorów mogą zostać zamknięte po inicjacji. Uwaga: BAM=Block Availability Map=Mapa dostępności bloków.

72 DISK FULL

Zapełnione są wszystkie bloki, albo nie ma już miejsca w katalogu na wpisanie następnego zbioru. Uwaga: Błąd ten jest zgłaszany gdy na dyskietce są jeszcze dwa bloki wolne, aby móc zamknąć aktualnie otwarty zbiór.

73 DOS MISMATCH (73, CBM DOS V2.6 1541)

System DOS wersja 2 potrafi odczytywać dyskietki zapisane przez DOS 1, nie może jednak nic na nie zapisać ponieważ inny jest format tych dyskietek. Komunikat ten pojawia się przy każdej próbie zapisu na dyskietkę o niezgodnym formacie. (Dostępny jest program pozwalający na zamianę jednego formatu na drugi). Komunikat ten może się także pojawić po włączeniu stacji.

74 DRIVE NOT READY

Próba korzystania ze stacji dysków, gdy nie jest wsunięta dyskietka.

DODATEK C

BASIC COMMODORE 64

Instrukcja ta wprowadziła już podstawy języka BASIC, wystarczająco, aby poznać ogólne zasady programowania i część podstawowych komend. Niniejszy dodatek zawiera pełną listę komend i funkcji języka BASIC, łącznie z krótkimi opisami. Warto jest eksperymentować, wpisując kolejne komendy, patrząc jaki jest ich efekt. Należy pamiętać, że nie istnieje możliwość zniszczenia komputera jakąkolwiek instrukcją.

Dodatek ten podzielony jest na kolejne części, opisujące różne operacje w języku BASIC. Są to:

1. Zmienne i Operatory. Opis różnych rodzajów zmiennych i operatorów, dopuszczalnych nazw zmiennych.
2. Polecenia. Opis poleceń używanych do pracy z programami, edycji, wczytywania i zapisywania.
3. Komendy. Opis komend języka BASIC używanych w numerowanych liniach programu.
4. Funkcje. Opis funkcji numerycznych i łańcuchowych (znakowych).

ZMIENNE

Commodore 64 używa w języku BASIC trzech rodzajów zmiennych. Są to dwa rodzaje zmiennych numerycznych: rzeczywiste i całkowite, oraz zmienne łańcuchowe (znakowe). Nazwa zmiennej może składać się z pojedynczej litery, litery i cyfry lub dwóch liter. Zmienną całkowitą oznacza się przez dodanie znaku % za nazwą zmiennej, natomiast zmienną łańcuchową przez dodanie $.

PRZYKŁADY

Nazwy zmiennych rzeczywistych: A, A5, BZ
Nazwy zmiennych całkowitych: A%, A5%, BZ%
Nazwy zmiennych łańcuchowych: A$, A5$, BZ$

TABLICE są listami zmiennych tego samego typu, określonymi jedną nazwą. Kolejne elementy listy są wyróżniane indeksem. Wielkość tablicy definiuje się komendą DIM. Elementami tablic mogą być zmienne rzeczywiste, całkowite i łańcuchowe. Przy definiowaniu tablicy należy podać w nawiasie ilość jej elementów:

```
DIM A(7), BZ%(50), A$(11), PT(20,20)
```

UWAGA: Istnieją trzy nazwy zmiennych zarezerwowane do używania komputera, które nie mogą być definiowane przez program. Są to: TI, TI$ i ST. ST jest zmienną zawierającą status ostatniej operacji wejścia/wyjścia. Wartość ta zmienia się, jeżeli wystąpiły jakieś problemy podczas wczytywania danych z taśmy lub dysku.

Zmienne TI oraz TI$ są związane z wewnętrznym zegarem czasu rzeczywistego wbudowanym w C 64. Zmienna TI jest zwiększana o 1 co 1/60 sekundy. W momencie włączenia komputera ustawiana jest ona na wartość 0 i jej zawartość można zmienić tylko zmieniając zawartość zmiennej TI$.

Zmienna TI$ jest łańcuchem znaków, stale uaktualnianym przez system. Pierwsze dwa znaki zawierają liczbę godzin, znaki trzeci i czwarty określają liczbę minut, natomiast ostatnie dwa znaki (5 i 6) określają liczbę sekund. Zmiennej tej można nadać dowolną wartość i od tej chwili będzie ona uaktualniana przez system.

```
TI$="101530" ustawia zegar na godzinę 10:15 i 30 sekund.
```

Zawartość zegara przepada w momencie wyłączenia komputera, a po włączeniu zawartość jest zerowana.

OPERATORY

Komputer rozróżnia następujące operatory matematyczne:

+ Dodawanie,
- Odejmowanie,
* Mnożenie,
/ Dzielenie,
^ Potęgowanie.

Jeżeli w linii występuje kilka operatorów, działania wykonywane są według pewnej kolejności. Najpierw potęgowanie, potem mnożenie i dzielenie, a na końcu dodawanie i odejmowanie.

Można zmienić kolejność wykonywania obliczeń ujmując w nawiasy wyrażenia, które mają być wykonane jako pierwsze.

Ponadto występują następujące operatory relacji:
= Równe
< Mniejsze niż
> Większe niż
<= Mniejsze lub równe
>= Większe lub równe
<> Różne

I dodatkowo operatory logiczne

AND
OR
NOT

Z operatorów tych korzysta się przy łączeniu kilku warunków w komendach IF ... THEN. Na przykład:

IF A=B AND C=D THEN 100 (Wymaga aby oba warunki były prawdziwe)

IF A=B OR C=D THEN 100 (Wystarczy aby jeden warunek był prawdziwy)

POLECENIA

CONT

Polecenie to pozwala na podjęcie wykonywania programu dokładnie w miejscu, w którym jego wykonywanie zostało przerwane naciśnięciem klawisza STOP (ale nie łącznie STOP i RESTORE), komendą STOP lub END w programie.
Polecenie CONT nie będzie mogło zadziałać, gdy została zmieniona lub dodana linia programu. Pojawi się wtedy komunikat CAN'T CONTINUE ERROR.

LIST

Polecenie LIST pozwala na obejrzenie linii programu BASIC znajdujących się w pamięci komputera. Można przeglądać cały program lub jego wybrany fragment.

LIST	Pokazuje cały program.
LIST 10 -	Pokazuje linie od 10 do końca.
LIST 10	Pokazuje tylko linię 10.
LIST - 10	Pokazuje linie od początku do linii 10.
LIST 10 - 20	Pokazuje linie od 10 do 20.

LOAD

Polecenie to służy do wczytania programu z taśmy lub dysku do pamięci komputera tak, aby móc z niego korzystać. Wpisanie samego polecenia LOAD powoduje wczytanie z magnetofonu pierwszego napotkanego programu. Dodanie za tym napisem nazwy programu ujętej w cudzysłów powoduje poszukiwanie programu o danej nazwie. Po nazwie można dodać jeszcze przecinek i liczbę. Liczba ta określa numer urządzenia, z którego należy wczytywać program.
Jeżeli nie podano numeru urządzenia komputer przyjmuje domyślną wartość 1, oznaczającą magnetofon. Innym urządzeniem zazwyczaj wykorzystywanym jest stacja dysków, której odpowiada numer 8.

LOAD	Odczytuje pierwszy program z taśmy.
LOAD "HALO"	Szuka na taśmie programu nazwanego HALO i jeśli go znajdzie, wczytuje go do pamięci.
LOAD A$	Szuka programu o nazwie określonej zmienną A$
LOAD "HALO",8	Szuka programu nazwanego HALO na dysku.
LOAD "*",8	Wczytuje pierwszy program na dysku.

Jeżeli program (zazwyczaj w kodzie maszynowym) ma być wczytany w określone miejsce w pamięci należy dodać tzw. parametr wtórny równy 1.

LOAD "K/M PROG",1,1 Wczytuje program z magnetofonu bez przemieszczania go w pamięci.

NEW

Polecenie to usuwa z pamięci program i wszystkie zmienne, które były używane. Program przepadnie jeżeli nie był zapisany poleceniem SAVE. NALEŻY BYĆ OSTROŻNYM GDY UŻYWA SIĘ TEGO POLECENIA.

NEW może być użyte także jako komenda w linii programu. Gdy program dochodzi do linii zawierającej tę komendę, zostaje usunięty z pamięci.

RUN

Polecenie to powoduje wykonywanie programu znajdującego się w pamięci. Jeżeli po tym poleceniu nie wpisano żadnego numeru linii program rozpocznie się od linii o najniższym numerze. Podanie numeru powoduje rozpoczęcie programu od wskazanej linii.

RUN Rozpoczyna wykonywanie programu od pierwszej linii.
RUN 50 Rozpoczyna wykonywanie programu od linii 50
RUN X Pokaże się napis UNDEF'D STATEMENT ERROR. Należy zawsze podać liczbę określającą numer linii, a nie zmienną.

SAVE

Polecenie to zapisuje program znajdujący się w pamięci komputera na taśmę lub dysk. Wpisanie tylko SAVE i naciśnięcie klawisza RETURN spowoduje zapisanie programu na taśmę. Komputer nie ma możliwości sprawdzenia co znajduje się na tej taśmie; więc należy uważać aby nie skasować poprzednich programów.

Jeżeli za słowem SAVE podano nazwę programu ujętą w cudzysłów lub zmienną łańcuchową, komputer nada tę nazwę zapisywanemu programowi, co ułatwi potem odnalezienie tego programu.

Za nazwą programu można podać numer urządzenia, na które program ma być zapisany oraz kolejny przecinek i parametr wtórny. Wartość 1 parametru wtórnego oznacza, że zapisywany właśnie program będzie wczytywany w to samo miejsce w pamięci, które zajmuje obecnie. Podanie liczby 2 powoduje zapisanie znacznika końca taśmy (End-Of-Tape marker). Gdy podczas szukania innego programu komputer spotka ten znacznik poszukiwanie zostaje przerwane i pojawia się komunikat FILE NOT FOUND (programu nie znaleziono). Można też podać wartość 3 oznaczającą jednocześnie koniec taśmy i znacznik wczytywania programu w to samo miejsce.

SAVE Zapisuje program bez nazwy na taśmę.
SAVE "HALO" Zapisuje na taśmę pod nazwę HALO.
SAVE A$ Zapisuje na taśmę pod nazwą określoną w A$.
SAVE "HALO",8 Zapisuje na dysk.
SAVE "HALO",1 ,1 Zapisuje na taśmę pod nazwą HALO i z zaznaczeniem, że program będzie wczytywany w to samo miejsce pamięci.
SAVE "HALO",1,2 Zapisuje na taśmę pod nazwą HALO i ze znacznikiem końca taśmy.
SAVE "HALO",1,3 j/w z zaznaczeniem wczytywania w to samo miejsce w pamięci.

VERIFY

Polecenie to służy do sprawdzenia poprawności ostatniego zapisu programu. Porównuje ono zawartość wskazanego zapisu z zawartością pamięci komputera.

VERIFY Porównuje z najbliższym programem na taśmie.
VERIFY "HALO" Szuka na taśmie programu HALO i porównuje go z pamięcią.
VERIFY "HALO",8 Szuka na dysku programu HALO i porównuje go z pamięcią.

Jeżeli pokaże się komunikat VERIFY ERROR, oznacza to błąd przy zapisie. Należy program zapisać ponownie. Błąd ten powstanie też jeżeli po zapisie została zmieniona jakakolwiek linia programu.

KOMENDY

CLOSE

Komenda ta powoduje zamknięcie zbioru otwartego komendą OPEN. Numer podany po tej komendzie określa numer zbioru, który ma zostać zamknięty.

CLOSE 2 Zamyka zbiór numer 2.

CLR

Komenda ta usuwa z pamięci komputera wszystkie zmienne, a pozostawia tylko program. Komenda ta jest wykonywana automatycznie po wydaniu polecenia RUN.

CMD

Komenda CMD powoduje skierowanie danych na drukarkę lub do zbioru na dysku zamiast na ekran. Oznacza to, że wszystkie komendy takie jak PRINT czy LIST zamiast wyświetlać linie na ekranie wydrukują je np. na drukarce (nie dotyczy to komend POKE wpisujących dane bezpośrednio do pamięci ekranu). Po słowie CMD należy podać numer zbioru otwartego komendą OPEN.

```
OPEN 1,4  Otwiera zbiór w urządzeniu 4, tj. drukarce.
CMD 1     Skierowuje dane do zbioru.
LIST      Program zostanie wypisany na drukarce, a nie na ekranie.
```

Aby ponownie wyświetlać napisy na ekranie trzeba zamknąć zbiór komendą CLOSE 1.

DATA

Po tej komendzie wypisane są dane, które będą odczytywane komendą READ. Dane te mogą być liczbami albo łańcuchami oddzielonymi od siebie przecinkami. Łańcuchy znaków nie muszą być ujęte w cudzysłów o ile nie zawierają znaków spacji, przecinków lub dwukropków. Jeżeli nic nie jest wpisane między kolejnymi przecinkami komenda READ wczyta zero jako liczbę lub pusty łańcuch.

```
DATA 12, 17, "RAZ, DWA", 3.14, JANUSZ
```

DEF FN

Jest to komenda pozwalająca zdefiniować własną funkcję matematyczną. Pozwala to na zastąpienie powtarzającego się skomplikowanego wzoru przez nazwę funkcji. Nazwa funkcji musi spełniać te same założenia co nazwa zmiennej (maks. 2 znaki). Po komendzie DEF FN i nazwie funkcji musi znajdować się w nawiasie zmienna liczbowa, a następnie znak = i wzór. Później w programie można odwoływać się do tego wzoru pisząc słowo FN nazwę funkcji oraz liczbę w nawiasie, która będzie podstawiona zamiast zmiennej.

```
10 DEF FN A(X) = 12*(34.75 - X/3)
20 PRINT FN A(7)
```

W tym wypadku wynik jest 137.

DIM

Komenda DIM definiuje i przydziela miejsce w pamięci dla tablic. Należy mieć na uwadze, że tablice zajmują całą pamięć niezbędną do pomieszczenia wszystkich zmiennych i nie deklarować dużo większych rozmiarów niż jest to potrzebne. Należy też pamiętać, że indeksy tablic zaczynają się od 0, tak więc, aby obliczyć ilość elementów należy dodać 1 do wymiaru tablicy.

```
DIM A$(40)        41 elementów,
DIM B7(15)        16 elementów,
DIM CC%(4,4,4)    125 elementów.
```

Wielkość tablicy można zdefiniować tylko raz. Uwaga: odwołanie się do elementu tablicy przed jej zdefiniowaniem powoduje automatyczne zadeklarowanie wielkości DIM X(10) tj. jedenaście elementów.

END

Komenda END powoduje natychmiastowe zakończenie wykonywania programu tak jak przejście za ostatnią linię programu. Polecenie CONT powoduje ponowne wykonywanie programu od pierwszej linii.

FOR .. TO .. STEP

Komenda ta, łącznie z komendą NEXT, powoduje wykonanie pewnego ciągu instrukcji określoną ilość razy. Format komendy:

```
FOR <naz.zm.>=<wart.pocz.> TO <wart.końc.> STEP <krok>
```

Zmiennej <naz.zm.> przypisywana jest wartość początkowa, a następnie po każdym wykonaniu instrukcji zawartych w pętli zwiększana jest ona o <krok>. Pętla jest wykonywana aż zmienna nie przekroczy wartości końcowej. Jeżeli pominięto słowo STEP i ostatnią liczbę, jako <krok> przyjmowana jest wartość 1.

```
10 FOR L=1 TO 3 STEP 0.1
20 PRINT L
30 NEXT
```

Jeżeli po słowie STEP jest liczba ujemna, wartość początkowa powinna być większa od końcowej.

GET

Komenda GET pozwala na wczytanie stanu klawiatury. Przypisuje ona wymienionej zmiennej znak wciśniętego klawisza lub pusty łańcuch, jeżeli nie został wciśnięty żaden klawisz. Jeżeli zmienna podana za GET jest typu numerycznego i naciśnięto inny klawisz niż z liczbą, program zostanie przerwany i wyświetlony będzie komunikat o błędzie. Aby program zatrzymał się do czasu naciśnięcia dowolnego klawisza należy wykonać pętlę:

```
10 GET A$: IF A$="" THEN 10
```

GET#

Komenda GET# służy do wczytania jednego znaku ze zbioru otwartego przedtem instrukcją OPEN.

```
GET#1,A$
```

GOSUB

Komenda ta działa podobnie jak komenda GOTO, z tym, że komputer pamięta numer linii, w którym ona wystąpiła. Jeżeli napotkana potem będzie komenda RETURN, program powróci do następnej instrukcji po GOSUB. Komenda ta jest przydatna gdy chce się z różnych miejsc w programie wywoływać ten sam ciąg instrukcji.

```
10 GOSUB 100: PRINT "PIERWSZY RAZ"
20 GOSUB 100: PRINT "DRUGI RAZ"
30 END
100 PRINT "ALA MA ASA":RETURN
```

GOTO lub GO TO

Napotkanie przez program komendy GOTO powoduje przejście do linii programu oznaczonej liczbę po tej komendzie.

IF .. TREN

Komenda ta pozwala wykonać jakieś działanie jeżeli spełniony jest warunek określony po słowie IF. Może to być dowolna komenda języka BASIC. Warunek może być porównaniem dwóch wyrażeń (porównanie operatorami = < > <= >= <>), lub pojedynczym wyrażeniem matematycznym. W drugim przypadku warunek jest spełniony gdy wyrażenie jest różne od zera. Można łączyć kilka warunków operatorami AND i OR lub stosować negację NOT.

```
IF X>10 THEN END
IF Y THEN PRINT "Y JEST ROZNE OD ZERA”
IF NOT Z AND (X>7) THEN 30
```

Podanie samej liczby po THEN w ostatnim przypadku jest równoznaczne z komendą GOTO 30.

INPUT

Komenda INPUT pozwala podawać komputerowi pewne dane w trakcie wykonywania programu. Program zatrzymuje się, wyświetla na ekranie znak zapytania (?) i oczekuje na wprowadzenie danych zakończone naciśnięciem klawisza RETURN. Po słowie INPUT należy podać zmienną lub listę zmiennych oddzielonych przecinkami, na które oczekuje program. Można też podać napis jaki pojawi się na ekranie ujmując go w cudzysłów i umieszczając przed listą zmiennych. Jeżeli po komendzie INPUT jest więcej niż jedna zmienna, należy je wpisywać oddzielając przecinkiem.

```
PRINT "PODAJ SWOJE IMIE";:INPUT A$
INPUT "PODAJ SWOJ KOD";B
```

INPUT#

INPUT# działa podobnie jak INPUT, ale dane odczytuje nie z klawiatury lecz ze zbioru.

```
INPUT#1,A
```

Uwaga: Aby odczytać ze zbioru takie znaki jak przecinek, cudzysłów czy kod klawisza RETURN, należy korzystać z komendy GET#.

LET

Komendę tę rzadko spotyka się w programach, gdyż można ją pominąć, jednakże jest to jedna z podstawowych komend języka BASIC, nazywana komendą przypisania.

```
LET X=10
LET D$="HALO"
```

NEXT

NEXT jest używane zawsze łącznie z komendą FOR .. TO. Służy ona do oznaczenia końca pętli otwartej komendą FOR. Za komendą NEXT może być wymieniona zmienna lub lista zmiennych oddzielonych przecinkami. Jeżeli po komendzie NEXT jest więcej niż jedna pętla to zamykana jest taka ilość pętli, ile jest zmiennych.

```
10 FOR X=1 TO 3
20 FOR Y=1 TO 2
30 PRINT X,Y
40 NEXT Y,X
```

ON

Komenda ON wykorzystywana jest łącznie z GOTO lub GOSUB jako odmiana komendy IF. Po słowie ON występuje wyrażenie, które jest obliczane i jeżeli wynik równy jest 1 wykonana zostanie pierwsza linia z listy, jeżeli wynik jest 2 - druga itd. Jeżeli wynik jest ujemny, równy zero lub większy niż ilość elementów listy, wykonana zostanie następna instrukcja po ON.

```
10 INPUT X
20 ON X GOSUB 100,110,120,130
```

OPEN

Komenda OPEN pozwala C 64 na dostęp do urządzeń jakimi są: stacja dysków, drukarka czy nawet ekran. Po słowie OPEN należy podać liczbę (0-255), określającą numer otwieranego zbioru, oraz numer urządzenia.

Numery urządzeń:

```
0 Klawiatura
1 Magnetofon
3 Ekran
4 Drukarka
8 Dysk
```

Po numerze urządzenia czasem należy podać tzw. parametr wtórny. W przypadku stacji dysków numer ten określa, który kanał ma być wykorzystany do przesyłania danych. W przypadku drukarki określa dodatkowe parametry druku np. pismo powiększone.

```
OPEN 1,0        Otwiera dostęp do ekranu
OPEN 2,8,2,"D"  Otwiera do czytania zbiór D na dysku
OPEN 3,4        Otwiera dostęp do drukarki
OPEN 15,8,15    Otwiera kanał poleceń dla dysku.
```

Więcej informacji na temat obsługi zbiorów znajduje się w Dodatku S.

POKE

Po komendzie POKE zawsze następują dwa parametry. Pierwszy określa miejsce w pamięci (adres), natomiast drugi wartość (0-255). Komenda POKE powoduje wpisanie w dane miejsce pamięci określonej wartości.

```
10 POKE 53281,0
20 S=4096*13
30 POKE S+29,8
```

PRINT

Komenda PRINT jest jedną z pierwszych komend, które poznaje się podczas nauki języka BASIC. Za komendą PRINT mogą znajdować się:

Łańcuch znaków ujęty w cudzysłów
Nazwa zmiennej
Funkcja lub wyrażenie matematyczne
Znak formatujący wydruk

Znaki formatujące wydruk to przecinek (,) lub średnik (;). Przecinek powoduje wyświetlenie napisu w najbliższej z czterech stref o szerokości 10 znaków, natomiast średnik powoduje wyświetlenie napisu bezpośrednio za poprzednim. Każdy z tych znaków może znajdować się na końcu linii. Ma on wtedy wpływ na następną komendę PRINT w programie.

```
10 PRINT "HALO"
20 PRINT "HALO"„ A$
30 PRINT A+B
40 PRINT J;
50 PRINT A,B,C,SIN(D)
```

Patrz także: funkcje POS, SPC, TAB.

PRINT#

Istnieje kilka różnic między PRINT a PRINT#. Po komendzie PRINT# występuje numer określający zbiór, do którego mają być zapisywane dane. Zbiór, do którego chce się coś zapisać musi być przedtem otwarty komendą OPEN. Przecinek i średnik mają taki sam efekt jak przy komendzie PRINT. Niektóre urządzenia mogą nie interpretować odpowiednio funkcji TAB i SPC.

```
10 PRINT#1, "WARTOSCI DANYCH:";A%,B1,C$
```

READ

Komenda READ przyporządkowuje kolejnym zmiennym dane pobrane z linii z komendą DATA. Należy zwrócić uwagę, aby READ nie odczytał zmiennej znakowej, gdy oczekuje na liczbę, gdyż spowoduje to wystąpienie błędu.

REM

Komenda REM służy do umieszczania w tekście programu komentarzy odnośnie danej linii, czy też całego programu. Można po tej komendzie wpisać dowolny ciąg znaków. Wszystkie znaki znajdujące się za komendą REM są ignorowane przez komputer. Uwaga: Ignorowany jest także dwukropek, tak więc za tą komendą nie można już dodać w tej linii żadnej innej komendy.

RESTORE

Wykonanie tej komendy powoduje przestawienie wskaźnika, określającego która dana ma być wczytana najbliższą instrukcją READ, na pierwszą linię DATA w programie lub, jeżeli podano numer linii, na tę linię.

RETURN

Komenda ta działa w połączeniu z komendą GOSUB. Gdy program dojdzie do tej komendy, następna wykonana będzie kolejna instrukcja po ostatnio realizowanej komendzie GOSUB. Jeżeli nie było przedtem komendy GOSUB pojawi się błąd RETURN WITHOUT GOSUB ERROR.

STOP

STOP powoduje przerwanie wykonywania programu i wyświetlenie komunikatu BREAK IN xxx, gdzie xxx jest linią, w której umieszczono komendę STOP. Można kontynuować wykonywanie programu z miejsca, w którym został on przerwany komendą CONT. STOP jest używane głównie do śledzenia programu.

SYS

Po komendzie SYS należy podać liczbę lub wyrażenie z zakresu od 0 do 65535. Jest to wywołanie programu napisanego w kodzie maszynowym. Od funkcji USR różni się tym, że nie pozwala na przekazywanie parametrów.

WAIT

Komenda ta powoduje zatrzymanie wykonywania programu i oczekiwanie, aż dana komórka pamięci komputera zmieni się w określony sposób. Po słowie WAIT należy podać adres tej komórki pamięci i jeden lub dwa parametry.

```
WAIT X,Y,Z
```

Komenda pobiera najpierw zawartość pamięci pod adresem X, następnie wykonuje operację logiczną EX-OR z parametrem Z (jeżeli jest podany), a potem logiczne AND z Y. Jeżeli wynik jest równy 0, operacja powtarza się, jeżeli nie - program wykonuje następną instrukcję.

FUNKCJE NUMERYCZNE

ABS(X) (wartość absolutna)
ABS oblicza wartość absolutną parametru.

ATN(X) (arcus tangens)
ATN oblicza kąt mierzony w radianach, którego tangens jest równy X.

COS(X) (cosinus)
COS oblicza wartość cosinus X, gdzie X jest mierzony w radianach.

EXP(X)
EXP oblicza wartość liczby e (2.71828183) do potęgi X.

FNxx(X)
Oblicza wartość funkcji zdefiniowanej przez programistę komendą DEF FNxx(X).

INT(X)
Oblicza wartość całkowitą liczby X, tzn. obcina wszystkie liczby po przecinku dziesiętnym. Wartość ta będzie mniejsza (lub równa) wartości X. Wartości liczb ujemnych będą również mniejsze od X.

LOG(X) (logarytm naturalny)
Oblicza wartość logarytmu naturalnego (przy podstawie e patrz funkcja EXP). Aby uzyskać wartość logarytmu przy innej podstawie Y należy wynik podzielić przez LOG(Y).

PEEK(X)
Podaje zawartość pamięci (zakres 0-255) pod adresem wskazanym przez X (zakres 0-65535). Funkcja ta jest często wykorzystywana w połączeniu z komendą POKE.

RND(X) (liczba losowa)
RND podaje losową liczbę z zakresu 0 - 1. Pierwsze użycie funkcji powinno mieć postać RND(-TI), aby rozpoczynać pracę generatora liczb losowych z różnych pozycji. Później X powinno mieć wartość 1 lub inną wartość dodatnią. Gdy X=0, funkcja będzie podawała ciągle tą samą wartość. Ujemna wartość X inicjuje generator. Używając tej samej liczby ujemnej uzyska się ciąg takich samych liczb "losowych". Wzór na generowanie liczb losowych z zakresu między X i Y

```
N=RND(1)*(Y-X)+X
```

gdzie Y jest górnym ograniczeniem, a X dolnym.

SGN(X) (znak)
SGN podaje wartość znaku liczby X, tzn. 1 - gdy X jest dodatnie, -1 gdy ujemne i 0 gdy X jest zerem.

SIN(X) (sinus)
SIN oblicza wartość sinus liczby X, gdzie X jest wyrażone w radianach.

SQR(X) (pierwiastek kwadratowy)
Funkcja ta oblicza wartość pierwiastka kwadratowego liczby X, gdzie X jest dodatnie lub równe zero. Gdy X jest ujemne wystąpi błąd ILLEGAL QUANTITY ERROR.

TAN(X) (tangens)
Oblicza tangens X, gdzie X jest wyrażone w radianach.

USR(x)
Wywołanie tej funkcji powoduje skok do programu w kodzie maszynowym, którego początek określony jest w pewnym miejscu pamięci. Parametr X jest przekazywany do tej procedury, która podaje inną wartość po zakończeniu wykonywania.

FUNKCJE ZNAKOWE

ASC(X$)
Funkcja ta podaje kod ASCII pierwszego znaku X$.

CHR$(X)
Funkcja ta jest przeciwna do ASC i podaje znak, którego kodem ASCII jest X.

LEFT$(X$,X)
Podaje łańcuch zawierający X znaków z lewej strony X$.

LEN(X$)
Podaje liczbę znaków, z których składa się X$.

MID$(X$,S,X)
Podaje łańcuch X znaków z X$, zaczynając od znaku na pozycji S

RIGHT$(X$,X)
Podaje łańcuch zawierający X znaków z prawej strony X$.

STR$(X)
Podaje łańcuch znaków identyczny z tym, jaki pokazałby się na ekranie po komendzie PRINT.

VAL(X$)
Jest to funkcja odwrotna do STR$. Przekształca ona łańcuch znaków na liczbę. Funkcja ta bada łańcuch od strony lewej do prawej aż dojdzie do końca łańcucha lub znaku nie będącego cyfrą.

```
X=VAL(" 123.456")      X=123.456
X=VAL("12A13B")        X=12
X=VAL("RIU017")        X=0
X=VAL("-1.23.45.67")   X=-1.23
```

POZOSTAŁE FUNKCJE

FRE(X)
Funkcja ta oblicza ilość wolnej pamięci w komputerze niezależnie od wartości X. Jeżeli ilość ta wynosi powyżej 32 kB, podawana jest wielkość ujemna.

POS(X)
Podaje numer kolumny (0-39), od której będzie wyświetlany następny napis komendę PRINT. X nie jest używane i może mieć dowolną wartość.

SPC(X)
Używane jest w komendzie PRINT aby przesunąć wydruk o X znaków w prawo.

TAB(X)
Również używane w komendzie PRINT. Następny tekst będzie wyświetlany od kolumny X.

DODATEK D

SKRÓTY KOMEND BASIC

Aby oszczędzić czasu podczas pisania programów dla Commodore 64 BASIC pozwala na stosowanie skrótów przy zapisie większości komend. Skrótem komendy PRINT jest znak zapytania. Skróty pozostałych komend uzyskuje się wpisując początkową jedną lub dwie litery komendy, a potem następną trzymając jednocześnie wciśnięty klawisz SHIFT. Jeżeli komendy zostały zapisane skrótami w programie, po poleceniu LIST ukażą się pełne nazwy.

Command	Abbreviation	Looks like this on screen	Command	Abbreviation	Looks like this on screen
ABS	A SHIFT B	A	END	E SHIFT N	E
AND	A SHIFT N	A	EXP	E SHIFT X	E
ASC	A SHIFT S	A	FN	NONE	FN
ATN	A SHIFT T	A	FOR	F SHIFT O	F
CHR$	C SHIFT H	C	FRE	F SHIFT R	F
CLOSE	CL SHIFT O	CL	GET	G SHIFT E	G
CLR	C SHIFT L	C	GET#	NONE	GET#
CMD	C SHIFT M	C	GOSUB	GO SHIFT S	GO
CONT	C SHIFT O	C	GOTO	G SHIFT O	G
COS	NONE	COS	IF	NONE	IF
DATA	D SHIFT A	D	INPUT	NONE	INPUT
DEF	D SHIFT E	D	INPUT#	I SHIFT N	I
DIM	D SHIFT I	D	INT	NONE	INT

Command	Abbreviation	Looks like this on screen
LEFT$	LE SHIFT F	LE
LEN	NONE	LEN
LET	L SHIFT E	L
LIST	L SHIFT I	L
LOAD	L SHIFT O	L
LOG	NONE	LOG
MID$	M SHIFT I	M
NEW	NONE	NEW
NEXT	N SHIFT E	N
NOT	N SHIFT O	N
ON	NONE	ON
OPEN	O SHIFT P	O
OR	NONE	OR
PEEK	P SHIFT E	P
POKE	P SHIFT O	P
POS	NONE	POS
PRINT	?	?
PRINT#	P SHIFT R	P
READ	R SHIFT E	R
REM	NONE	REM
RESTORE	RE SHIFT S	RE
RETURN	RE SHIFT T	RE

Command	Abbreviation	Looks like this on screen
RIGHT$	R SHIFT I	R
RND	R SHIFT N	R
RUN	R SHIFT U	R
SAVE	S SHIFT A	S
SGN	S SHIFT G	S
SIN	S SHIFT I	S
SPC(	S SHIFT P	S
SQR	S SHIFT Q	S
STATUS	ST	ST
STEP	ST SHIFT E	ST
STOP	S SHIFT T	S
STR$	ST SHIFT R	ST
SYS	S SHIFT Y	S
TAB(	T SHIFT A	T
TAN	NONE	TAN
THEN	T SHIFT H	T
TIME	TI	TI
TIME$	TI$	TI$
USR	U SHIFT S	U
VAL	V SHIFT A	V
VERIFY	V SHIFT E	V
WAIT	W SHIFT A	W

DODATEK E

KODY EKRANOWE

Poniższe zestawienie zawiera znaki, z których składa się zestaw znaków Commodore 64. Znając ich numery można sprawić, aby ukazany się na ekranie komendą POKE lub też sprawdzać zawartość ekranu komendę PEEK.

W C 64 dostępne są dwa zestawy znaków, jednakże w danym momencie na ekranie może pojawiać się tylko jeden z nich. Oznacza to, że nie można pokazać na ekranie jednocześnie kilku znaków z jednego i kilku z drugiego zestawu. Naciskając klawisze SHIFT i C= (COMMODORE KEY) przełącza się między jednym i drugim zestawem. Można także zmieniać zestaw znaków programowo komendami POKE 53272,21 (duże litery) lub POKE 53272,23 (małe litery).

Każdy znak, zarówno z pierwszego, jak i z drugiego zestawu może być wyświetlony w odwrotnych kolorach tła i liter (REVERSE). Aby tak go wyświetlić należy dodać do jego numeru liczbę 128.

Aby wyświetlić zapełnione kółko na ekranie na pozycji 1504 należy wpisać rozkaz POKE 1504,81.

Każdemu miejscu na ekranie odpowiada inne miejsce w pamięci, w którym przechowywana jest informacja o kolorach (adresy 55296 - 56295). Aby zmienić kolor kółka na ekranie na np. żółty (kod koloru 7), należy wpisać tę wartość w odpowiednie miejsce w pamięci (adres 55776) komendą POKE 55776,7.

Dodatek G zawiera mapy pamięci ekranu i kolorów pozwalające ustalić adresy danego punktu na ekranie.

Uwaga: Następujące znaki są identyczne w obu zestawach: 27-64, 91-93, 96-104, 106-121, 123-127.

SCREEN CODES

SET 1	SET 2	POKE	SET 1	SET 2	POKE	SET 1	SET 2	POKE
@		0	C	c	3	F	f	6
A	a	1	D	d	4	G	g	7
B	b	2	E	e	5	H	h	8

SET 1	SET 2	POKE	SET 1	SET 2	POKE	SET 1	SET 2	POKE
I	i	9	%		37		A	65
J	j	10	&		38		B	66
K	k	11	'		39		C	67
L	l	12	(		40		D	68
M	m	13	)		41		E	69
N	n	14	*		42		F	70
O	o	15	+		43		G	71
P	p	16	,		44		H	72
Q	q	17	–		45		I	73
R	r	18	.		46		J	74
S	s	19	/		47		K	75
T	t	20	0		48		L	76
U	u	21	1		49		M	77
V	v	22	2		50		N	78
W	w	23	3		51		O	79
X	x	24	4		52		P	80
Y	y	25	5		53		Q	81
Z	z	26	6		54		R	82
[		27	7		55		S	83
£		28	8		56		T	84
]		29	9		57		U	85
↑		30	:		58		V	86
←		31	;		59		W	87
SPACE		32	<		60		X	88
!		33	=		61		Y	89
"		34	>		62		Z	90
#		35	?		63			91
$		36			64			92

SET 1	SET 2	POKE	SET 1	SET 2	POKE	SET 1	SET 2	POKE
		93			105			117
		94			106			118
		95			107			119
SPACE		96			108			120
		97			109			121
		98			110			122
		99			111			123
		100			112			124
		101			113			125
		102			114			126
		103			115			127
		104			116			

Kody 128-255 są zamienionymi kolorami znakami o kodach 0-127.

DODATEK F

KODY ASCII I CHR$

Niniejszy dodatek informuje jaki znak ukaże się po komendzie PRINT CHR$(X), dla wszystkich możliwych wartości X. Podaje także wartości uzyskiwane komendę PRINT ASC("x") gdzie x jest dowolnym znakiem jaki można wpisać z klawiatury. Jest to przydatne do sprawdzania jaki znak uzyskano komendą GET, wykorzystania w komendzie PRINT znaków, których nie można ująć w cudzysłów (np. przełączanie między dużymi i małymi literami klawiszami SHIFT i C=).

PRINTS	CHR$	PRINTS	CHR$	PRINTS	CHR$	PRINTS	CHR$
	0	CRSR ↑	17	"	34	3	51
	1	RVS ON	18	#	35	4	52
	2	CLR HOME	19	$	36	5	53
	3	INST DEL	20	%	37	6	54
	4		21	&	38	7	55
WHT	5		22	'	39	8	56
	6		23	(	40	9	57
	7		24	)	41	:	58
DISABLES SHIFT C=	8		25	*	42	;	59
ENABLES SHIFT C=	9		26	+	43	<	60
	10		27	,	44	=	61
	11	RED	28	-	45	>	62
	12	CRSR →	29	.	46	?	63
RETURN	13	GRN	30	/	47	@	64
SWITCH TO LOWER CASE	14	BLU	31	0	48	A	65
	15	SPACE	32	1	49	B	66
	16	!	33	2	50	C	67

PRINTS	CHR$	PRINTS	CHR$	PRINTS	CHR$	PRINTS	CHR$
D	68		97		126	Grey 3	155
E	69		98		127		156
F	70		99		128		157
G	71		100	Orange	129		158
H	72		101		130		159
I	73		102		131	SPACE	160
J	74		103		132		161
K	75		104	f1	133		162
L	76		105	f3	134		163
M	77		106	f5	135		164
N	78		107	f7	136		165
O	79		108	f2	137		166
P	80		109	f4	138		167
Q	81		110	f6	139		168
R	82		111	f8	140		169
S	83		112		141		170
T	84		113	SWITCH TO UPPER CASE	142		171
U	85		114		143		172
V	86		115		144		173
W	87		116		145		174
X	88		117		146		175
Y	89		118		147		176
Z	90		119		148		177
[	91		120	Brown	149		178
£	92		121	Lt. Red	150		179
]	93		122	Grey 1	151		180
↑	94		123	Grey 2	152		181
←	95		124	Lt. Green	153		182
	96		125	Lt. Blue	154		183

PRINTS	CHR$	PRINTS	CHR$	PRINTS	CHR$	PRINTS	CHR$
	184		186		188		190
	185		187		189		191

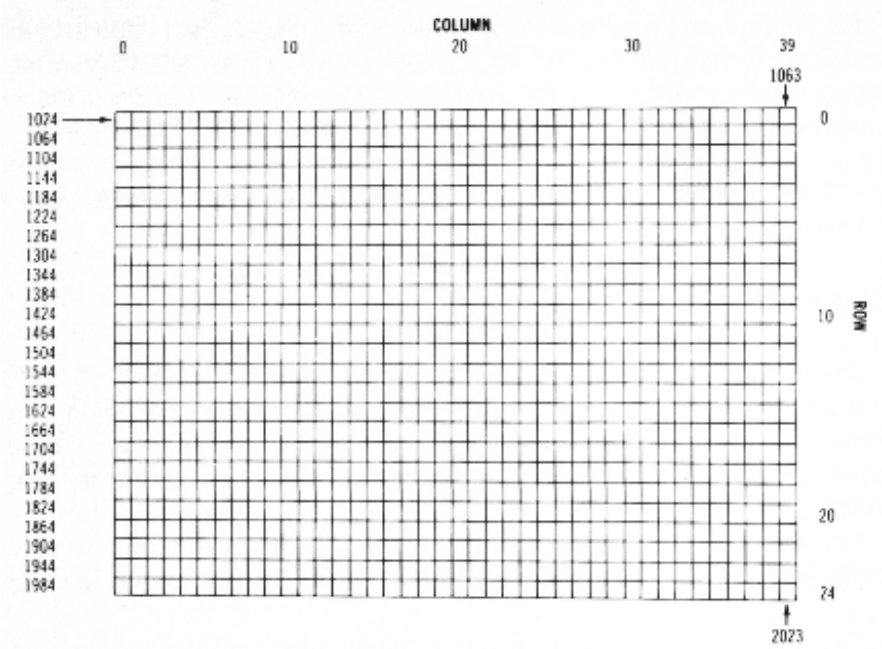
KODY	**192-223**	**TAKIE SAME JAK**	**96-127**
KODY	**224-254**	**TAKIE SAME JAK**	**160-190**
KOD	**255**	**TAKI SAM JAK**	**126**

DODATEK G

MAPY PAMIĘCI EKRANU I KOLORÓW

Siatka znajdująca się poniżej pokazuje położenie w pamięci (adresy) poszczególnych pól ekranu. Każdemu polu ekranu przypisane są dwa adresy. Zawartość jednego, opisanego w mapie pamięci ekranu, odpowiedzialna jest za kształt znaku tam wpisanego, natomiast zawartość drugiego, w mapie pamięci kolorów, określa kolor, w jakim ten znak ma być wyświetlany na ekranie.

MAPA PAMIĘCI EKRANU

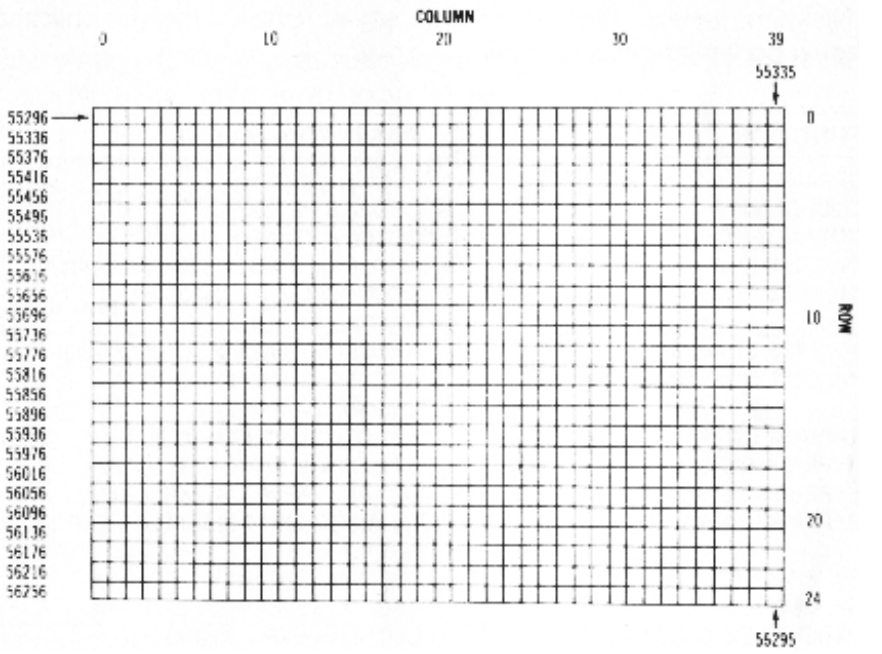


Aby zmienić kolor znaku do odpowiedniego miejsca w pamięci kolorów wpisać wartość:

0	CZARNY	8	POMARAŃCZOWY
1	BIAŁY	9	BRĄZOWY
2	CZERWONY	10	JASNOCZERWONY
3	NIEBIESKI (CYAN)	11	SZARY 1
4	PURPURA	12	SZARY 2
5	ZIELONY	13	JASNOZIELONY
6	NIEBIESKI	14	JASNONIEBIESKI
7	ŻÓŁTY	15	SZARY 3

Na przykład, aby zmienić kolor górnego lewego znaku na czerwony, należy wpisać POKE 55296,2.

MAPA PAMIĘCI KOLORÓW



DODATEK H

WZORY NA FUNKCJE MATEMATYCZNE

Niektóre funkcje niedostępne w języku BASIC można obliczać według podanych niżej wzorów.

FUNCTION	BASIC EQUIVALENT
SECANT	SEC(X)=1/COS(X)
COSECANT	CSC(X)=1/SIN(X)
COTANGENT	COT(X)=1/TAN(X)
INVERSE SINE	ARCSIN(X)=ATN(X/SQR(-X*X+1))
INVERSE COSINE	ARCCOS(X)= -ATN(X/SQR(-X*X+1)) + π/2
INVERSE SECANT	ARCSEC(X)=ATN(X/SQR(X*X-1))
INVERSE COSECANT	ARCCSC(X)=ATN(X/SQR(X*X-1)) + (SGN(X)-1*π/2)
INVERSE COTANGENT	ARCOT(X)=ATN(X) + π/2
HYPERBOLIC SINE	SINH(X)=(EXP(X)-EXP(-X)) /2
HYPERBOLIC COSINE	COSH(X)=(EXP(X)+EXP(-X)) /2
HYPERBOLIC TANGENT	TANH(X)=EXP(-X)/(EXP(X)+ EXP(-X)) *2+1
HYPERBOLIC SECANT	SECH(X)=2/(EXP(X)+EXP(-X))
HYPERBOLIC COSECANT	CSCH(X)= 2/(EXP(X)-EXP(-X))
HYPERBOLIC COTANGENT	COTH(X)=EXP(-X)/(EXP(X)-EXP(-X)) *2+1
INVERSE HYPERBOLIC SINE	ARCSINH(X)=LOG(X+SQR(X*X+1))
INVERSE HYPERBOLIC COSINE	ARCCOSH(X)=LOG(X+SQR(X*X-1))
INVERSE HYPERBOLIC TANGENT	ARCTANH(X)=LOG((1+X)/(1-X)) /2
INVERSE HYPERBOLIC SECANT	ARCSECH(X)=LOG((SQR(-X*X+1)+1/X)
INVERSE HYPERBOLIC COSECANT	ARCCSCH(X)=LOG((SGN(X)*SQR(X*X+1/X)
INVERSE HYPERBOLIC COTANGENT	ARCCOTH(X)=LOG((X+1)/(X-1)) /2

DODATEK I

UKŁAD WYPROWADZEŃ SYGNAŁÓW W GNIAZDACH KOMPUTERA

Niniejszy dodatek pokazuje, które sygnały, i w którym miejscu dostępne są poprzez zewnętrzne gniazda komputera.

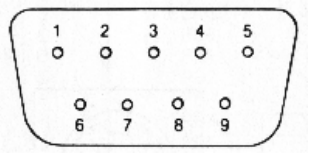
1) Gniazdo joysticka.
2) Gniazdo cartridge.
3) Audio/Video.
4) Wyjście szeregowe.
5) Wyjście TV.
6) Gniazdo magnetofonu.
7) User Port.

Control Port 1

Pin	Type	Note
1	JOY A0	
2	JOY A1	
3	JOY A2	
4	JOY A3	
5	POT AY	
6	BUTTON A/LP	
7	+5V	MAX. 100mA
8	GND	
9	POT AX	

Control Port 2

Pin	Type	Note
1	JOY B0	
2	JOY B1	
3	JOY B2	
4	JOY B3	
5	POT BY	
6	BUTTON B	
7	+5V	MAX. 100mA
8	GND	
9	POT BX	



Cartridge Expansion Slot

Pin	Type
1	GND
2	+5V
3	+5V
4	/IRQ
5	/CR/W
6	Dot Clock
7	I/O 1
8	/GAME
9	/EXROM
10	I/O 2
11	/ROML

Cartridge Expansion Slot

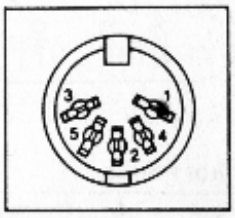
Pin	Type
12	BA
13	/DMA
14	CD7
15	CD6
16	CD5
17	CD4
18	CD3
19	CD2
20	CD1
21	CD0
22	GND

Pin	Type
A	GND
B	/ROMH
C	/RESET
D	/NMI
E	S 02
F	CA15
H	CA14
J	CA13
K	CA12
L	CA11
M	CA10

Pin	Type
N	CA9
P	CA8
R	CA7
S	CA6
T	CA5
U	CA4
V	CA3
W	CA2
X	CA1
Y	CA0
Z	GND

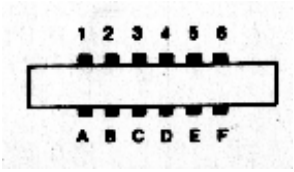
Audio/Video

Pin	Type
1	LUMINANCE
2	GND
3	AUDIO OUT
4	VIDEO OUT
5	AUDIO IN



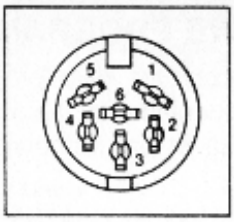
Cassette

Pin	Type
A-1	GND
B-2	+5V
C-3	CASSETTE MOTOR
D-4	CASSETTE READ
E-5	CASSETTE WRITE
F-6	CASSETTE SENSE



Serial I/O

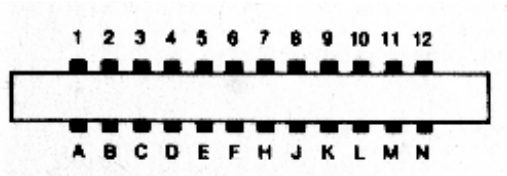
Pin	Type
1	SERIAL /SRQIN
2	GND
3	SERIAL ATN IN/OUT
4	SERIAL CLK IN/OUT
5	SERIAL DATA IN/OUT
6	/RESET



User I/O

Pin	Type	Note
1	GND	
2	+5V	MAX. 100mA
3	/RESET	
4	CNTI	
5	SP1	
6	CNT2	
7	SP2	
8	/PC2	
9	SER. ATN IN	
10	9VAC	MAX. 100mA
11	9VAC	MAX. 100mA
12	GND	

Pin	Type	Note
A	GND	
B	/FLAG2	
C	PB0	
D	PB1	
E	PB2	
F	PB3	
H	PB4	
J	PB5	
K	PB6	
L	PB7	
M	PA2	
N	GND	



DODATEK J

PRZYKŁADOWE PROGRAMY

Poniżej zamieszczono kilka programów, które można wpisać do komputera. Obserwując działanie tych programów, można poznać ogólne zasady pisania programów.

```
100 print"■jotto       jim butterfield"
120 input"■want instructions";z$:ifasc(z$)=78goto250
130 print"■try to guess the mystery 5-letter word"
140 print"■you must guess only legal 5-letter"
150 print"words, too..."
160 print"you will be told the number of matches"
170 print"(or 'jots') of your guess."
180 print"■hint: the trick is to vary slightly"
190 print"  from one guess to the next; so that"
200 print"  if you guess 'batch' and get 2 jots"
210 print"  you might try 'botch' or 'chart'"
220 print"  for the next guess..."
250 data bxbsf,ipccz,dbdif,esfbe,pggbm
260 data hpshf,ibudi,djwjm,kpmmz,lbzbl
270 data sbkbi,mfwfm,njnjd,boofy,qjqfs
280 data rvftu,sjwfs,qsftt,puufs,fwfou
290 data xfbwf,fyupm,nvtiz,afcsb,gjaaz
300 data uijdl,esvol,gmppe,ujhfs,gblfs
310 data cppui,mzjoh,trvbu,hbvaf,pxjoh
320 data uisff,tjhiu,bymft,hsvnq,bsfob
330 data rvbsu,dsffq,cfmdi,qsftt,tqbsl
340 data sbebs,svsbm,tnfmm,gspxo,esjgu
400 n=50
410 dim n$(n),z(5),y(5)
420 for j=1ton:readn$(j):nextj
430 t=ti
440 t=t/1000:ift>=1thengoto440
450 z=rnd(-t)
500 g=0:n$=n$(rnd(1)*n+1)
510 print "■i have a five letter word:":ifr>0goto560
520 print "guess (with legal words)"
530 print "and i'll tell you how many"
540 print "'jots', or matching letters,"
550 print "you have...."
560 g=g+1:input "your word";z$
570 if len(z$)<>5thenprint"you must guess a
    5-letter word!":goto560
580 v=0:h=0:m=0
590 forj=1to5
600 z=asc(mid$(z$,j,1)):y=asc(mid$(n$,j,1))-1:ify=64theny=90
610 ifz<65orz>90thenprint"that's not a word!":goto560
620 ifz=65orz=69orz=73orz=79orz=85orz=89thenv=v+1
630 ifz=ythenm=m+1
640 z(j)=z:y(j)=y:nextj
650 ifm=5goto800
660 ifv=0orv=5thenprint"come on..what kind of
    a word is that?":goto560
670 for j=1to5:y=y(j)
680 for k=1to5:ify=z(k)thenh=h+1:z(k)=0:goto700
690 next k
700 next j
710 print"■■■■■■■■■■■■■■■■■■■■";H;"JOTS"
720 ifg<30goto560
730 print"i'd better tell you.. word was '";
```

```
740 forj=1to5:printchr$(y(j));:nextj
750 print"'":goto810
800 print"you got it in only";g;"guesses."
810 input"another word";z$
820 r=1:ifasc(z$)<>78goto500


1 rem *** sequence
2 rem
3 rem *** from pet user group
4 rem *** software exchange
5 rem *** po box 371
6 rem *** montgomeryville, pa 18936
7 rem
50 dim a$(26)
100 z$="abcdefghijklmnopqrstuvwxuz"
110 z1$="12345678901234567890123456"
200 print"enter length of string to be sequenced"
220 input "maximum length is 26 ";s%
230 if s%<1 or s%>26 then 200
240 s=s%
300 for i=1 to s
310 a$(i)=mid$(z$,i,1)
320 next i
400 rem randomize string
420 for i=1 to s
430 k=int(rnd(1)*s+1)
440 t$=a$(i)
450 a$(i)=a$(k)
460 a$(k)=t$
470 next i
480 gosub 950
595 t=0
600 rem reverse substring
605 t=t+1
610 input "how many to reverse ";r%
620 if r%=0 goto 900
630 if r%>0 and r%<=s goto 650
640 print "must be between 1 and ";s: goto 610
650 r=int(r%/2)
660 for i=1 to r
670 t$=a$(i)
680 a$(i)=a$(r%-i+1)
690 a$(r%-i+1)=t$
700 next i
750 gosub 950
800 c=1: for i=2 to s
810 if a$(i)>a$(i-1) goto 830
820 c=0
830 next i
840 if c=0 goto 600
850 print "you did it in ";t;" tries"
900 rem check for another game
910 input "want to play again ";y$
920 if left$(y$,1)="y" or y$="ok" or y$="1" goto 200
930 end
950 print
960 print left$(z1$,s)
970 for i=1 to s: print a$(i);:next i
980 print "
990 return
```

This program courtesy of Gene Deals

```
 90 REM PIANO KEYBOARD
100 PRINT"▯ ▮ ▮ ▮ | ▮ ▮ ▮ | ▮ ▮ | ▮ ▮ "
110 PRINT" ▮ ▮ ▮ | ▮ ▮ ▮ | ▮ ▮ | ▮ ▮ "
120 PRINT" ▮ ▮ ▮ | ▮ ▮ ▮ | ▮ ▮ | ▮ ▮ "
130 PRINT" ▮ | | | | | | | | | | | | "
140 PRINT" ▮Q|W|E|R|T|Y|U|I|O|P|@|*|↑"
150 PRINT"▮'SPACE' FOR SOLO OR POLYPHONIC"
160 PRINT"▮'F1,F3,F5,F7' OCTAVE SELECTION"
170 PRINT"▮'F2,F4,F6,F8' WAVEFORM▮"
180 PRINT"HANG ON, SETTING UP FREQUENCY TABLE..."
190 S=13*4096+1024:DIMF(26):DIMK(255)
200 FORI=0TO28:POKES+I,0:NEXT
210 F1=7040:FORI=1TO26:F(27-I)=F1*5.3+30:F1=F1/2↑(1/12):NEXT
220 K$="@2WCEP5T6Y7U1900PQ-*€↑"
230 FORI=1TOLEN(K$):K(ASC(MID$(K$,I)))=I:NEXT
240 PRINT"▯                                                    "
250 AT=0:DE=0:SU=15:RE=9:SV=SU*16+RE:AV=AT*16+DE:
    WV=16:N=0:M=1:OC=4:HB=256:Z=7
260 FORI=0TO2:POKES+5+I*7,AT*16+DE:POKES+6+I*7,SU*16+RE
270 POKES+2+I*7,4000AND255:POKES+3+I*7,4000/256:NEXT
280 POKES+24,15:REM+16+64:POKES+23,7
300 GETA$:IFA$=""THEN300
310 FR=F(K(ASC(A$)))/M:T=V*7:CR=8+T+4:IFFR=ZTHEN500
320 POKES+6+T,Z:REM FINISH DEC/SUS
325 POKES+5+T,Z:REM FINISH ATT/REL
330 POKECR,8:POKECR,0:REM FIX OFF
340 POKES+T,FR-HB*INT(FR/HB):REM SET LO
350 POKES+1+T,FR/HB:REM SET HI
360 POKES+6+T,SV:REM SET DEC/SUS
365 POKES+5+T,AV:REM SET ATT/REL
370 POKECR,WV+1:FORI=1TO50*AT:NEXT
375 POKECR,WV:REM PULSE
380 IFP=1THENV=V+1:IFV=3THENV=0
400 GOTO300
500 IFA$="▮"THENM=1:OC=4:GOTO300
510 IFA$="▮"THENM=2:OC=3:GOTO300
520 IFA$="▮"THENM=4:OC=2:GOTO300
530 IFA$="▮"THENM=8:OC=1:GOTO300
540 IFA$="▮"THENW=0:WV=16:GOTO300
550 IFA$="▮"THENW=1:WV=32:GOTO300
560 IFA$="▮"THENW=2:WV=64:GOTO300
570 IFA$="▮"THENW=3:WV=128:GOTO300
580 IFA$=" "THENP=1-P:GOTO300
590 IFA$="▯"THEN200
600 GOTO300
300 PRINT"HIT A KEY"
310 GETA$:IFA$=""THEN310:WAIT FOR A KEY
320 PRINTA$:RETURN
```

NOTES:

Line 100 uses (SHIFT CLR/HOME), (CTRL 9),(CTRL]),(SHIFT B).
Line 150 uses (CRSR DOWN)
Line 240 uses (CRSR UP)
Line 500 uses (f1)
Line 510 uses (f3)
Line 520 uses (f5)
Line 530 uses (f7)
Line 540 uses (f2)
Line 550 uses (f4)
Line 560 uses (f6)
Line 570 uses (f8)
Line 590 uses (SHIFT CLR/HOME)

DODATEK K

ZAMIANA PROGRAMÓW NAPISANYCH W INNYCH DIALEKTACH JĘZYKA BASIC NA COMMODORE BASIC

Podczas wpisywania programów w języku BASIC przeznaczonych na inne komputery niż C 64, trzeba czasami dokonać pewnych drobnych poprawek. Oto niektóre przykłady.

Definiowanie długości łańcuchów.

Należy usunąć wszystkie komendy definiujące długość łańcuchów (DIM A$(I)). Komendy takie jak DIM A$(J,I) oznaczają deklarację J łańcuchów o długości I. Powinny być zastąpione komendę DIM A$(J).

Czasami do łączenia dwóch łańcuchów używa się przecinka lub znaku &. Znaki te należy wymienić na znak plus (+).

W języku BASIC na C 64 istnieją funkcje LEFT$, MID$ i RIGHT$ służące do wydzielenia fragmentu łańcucha znaków. Formy takie jak A$(n) oznacza n-ty znak łańcucha, a A$(n,m) oznacza ciąg m znaków począwszy od n-tego z A$. Polecenia dodania znaków do łańcucha należy zmieniać w/g zasady:

Inny BASIC	**Commodore BASIC**
A$(I) = X$	A$=LEFT$(A$,I-1)+X$+MID$(A$,I+1)
A$(I,J) = X$	A$=LEFT$(A$,I-1)+X$+MID$(A$,J+1)

Wielokrotne przypisanie.

Niektóre odmiany języka BASIC pozwalają na jednoczesne nadanie wartości dwóm zmiennym komendą:

```
10 LET B = C = 0
```

Commodore BASIC zinterpretowałby drugi znak = jako operator logiczny i nie zmieniłby wartości C, a gdyby C miało wartość 0 to zmiennej B przypisałby wartość -1. Takie wyrażenie należy zmienić na:

```
10 B = 0: C=0
```

Kilka instrukcji w jednej linii programu.

Niektóre odmiany języka BASIC oddzielają kolejne instrukcje w linii programu znakiem (\). Należy zmienić ten znak na dwukropek (:).

Funkcja MAT.

Programy używające funkcji MAT muszę zostać poprawione z wykorzystaniem pętli FOR ... NEXT.

DODATEK L

KOMUNIKATY BŁĘDÓW

Dodatek ten zawiera listę komunikatów o błędach pojawiających się podczas pracy komputera, z krótkim wyjaśnieniem ich przyczyn.

BAD DATA Ze zbioru wczytano łańcuch, gdy program oczekiwał liczby.
BAD SUBSCRIPT Program próbował odwołać się do elementu tablicy o numerze większym niż jej wielkość.
BREAK Program został zatrzymany naciśnięciem klawisza STOP lub komendę STOP.
CAN'T CONTINUE Polecenie CONT nie może działać, ponieważ albo program nie był przedtem uruchomiony poleceniem RUN, albo po zatrzymaniu zmieniono jakąś linię programu.
DEVICE NOT PRESENT Urządzenie, do którego odwołują się komendy OPEN, CLOSE, CMD, PRINT#, INPUT# lub GET# nie jest podłączone.
DIVISION BY ZERO Dzielenie przez zero jest niedozwolone.
EXTRA IGNORED Zbyt wiele danych podanych komendzie INPUT. Wczytywane jest tyle danych ile było zmiennych, reszta jest ignorowana.
FILE NOT FOUND Nie znaleziono programu o podanej nazwie.
FILE NOT OPEN Zbiór wskazany w komendzie CLOSE, PRINT#, GET#, INPUT# lub CMD nie został przedtem otwarty komendą OPEN.
FILE OPEN Próba ponownego otwarcia otwartego już zbioru.
FORMULA TOO COMPLEX Obliczane wyrażenie matematyczne jest za długie lub zawiera za dużo nawiasów. Powinno być podzielone na co najmniej dwie części.
ILLEGAL DEVICE NUMBER Próba niewłaściwego dostępu do urządzenia (np. próba czytania z ekranu lub drukarki).
ILLEGAL DIRECT Komenda INPUT może być użyta tylko w programie, a nie w trybie bezpośrednim.
LOAD Nastąpił błąd przy wczytywaniu programu z taśmy lub dysku.
MISSING FILE NAME Po komendzie LOAD lub SAVE dotyczącej dysku musi być podana nazwa programu.
NEXT WITHOUT FOR Program napotkał komendę NEXT nie wykonując przedtem komendy FOR lub zmienna podana po komendzie NEXT jest inna niż w komendzie FOR.
NOT INPUT FILE Próba czytania ze zbioru otwartego do zapisu.
NOT OUTPUT FILE Próba zapisu do zbioru otwartego do odczytu.
OUT OF DATA Wykonano komendę READ, a wszystkie dane z linii DATA zostały już wczytane.
OUT OF MEMORY Brak pamięci na program lub zmienne. Może pojawić się też jeżeli w programie występuje za dużo zagnieżdżonych pętli FOR lub komend GOSUB.
OVERFLOW Wynik obliczeń matematycznych jest większy niż największa dopuszczalna wartość tj. +/- 1.70141884 E+38.
REDIM'D ARRAY Wielkość tablicy można definiować tylko raz. Jeżeli w programie odwołano się do elementu tablicy przed wykonaniem komendy DIM, automatycznie tworzona jest tablica o 11 elementach i próba późniejszej definicji wielkości tej tablicy spowoduje wystąpienie tego błędu.
REDO FROM START Wpisano litery gdy komenda INPUT oczekiwała liczby. Należy ponownie podać liczbę. Program nie przerywa swojego działania.
RETURN WITHOUT GOSUB Program napotkał komendę RETURN, nie była wykonana przedtem komenda GOSUB.
STRING TOO LONG Łańcuch znaków (w tym linia programu) nie może być dłuższy niż 255 znaków.
?SYNTAX ERROR Komenda nie rozpoznana przez C 64. Brak nawiasów, źle wpisana komenda itp.
TOO MANY FILES Próbowano otworzyć więcej niż 10 zbiorów.
TYPE MISMATCH Błąd ten występuje gdy próbuje się podstawić liczbę zamiast łańcucha znaków i odwrotnie.
UNDEF'D FUNCTION Odwołano się do funkcji definiowanej przez użytkownika nie zdefiniowanej przedtem komendą DEF FN.
UNDEF'D STATEMENT Linia podana po komendzie GOTO, GOSUB lub RUN nie istnieje.
VERIFY Program zapisany na dysk lub taśmę różni się od programu w pamięci komputera.

DODATEK M

CZĘSTOTLIWOŚCI NUT

Dodatek ten zawiera wartości, jakie należy wpisać do rejestrów układu SID, aby uzyskać dźwięk odpowiadający danej nucie. HIGH oznacza starszy bajt, natomiast LOW młodszy.

			OSCILLATOR FREQ	
MUSICAL NOTE		Cycles/Sec.	HIGH	LOW
0	C-0	268	1	12
1	C#-0	284	1	28
2	D-0	301	1	45
3	D#-0	318	1	62
4	E-0	337	1	81
5	F-0	358	1	102
6	F#-0	379	1	123
7	G-0	401	1	145
8	G#-0	425	1	169
9	A-0	451	1	195
10	A#-0	477	1	221
11	B-0	506	1	250
16	C-1	536	2	24
17	C#-1	568	2	56
18	D-1	602	2	90
19	D#-1	637	2	125
20	E-1	675	2	163
21	F-1	716	2	204
22	F#-1	758	2	246
23	G-1	803	3	35
24	G#-1	851	3	83
25	A-1	902	3	134
26	A#-1	955	3	187
27	B-1	1012	3	244
32	C-2	1072	4	48
33	C#-2	1136	4	112
34	D-2	1204	4	180
35	D#-2	1275	4	251
36	E-2	1351	5	71
37	F-2	1432	5	152
38	F#-2	1517	5	237
39	G-2	1607	6	71
40	G#-2	1703	6	167
41	A-2	1804	7	12
42	A#-2	1911	7	119
43	B-2	2025	7	233
48	C-3	2145	8	97
49	C#-3	2273	8	225
50	D-3	2408	9	104
51	D#-3	2551	9	247
52	E-3	2703	10	143
53	F-3	2864	11	48
54	F#-3	3034	11	218
55	G-3	3215	12	143
56	G#-3	3406	13	78
57	A-3	3608	14	24
58	A#-3	3823	14	239
59	B-3	4050	15	210
64	C-4	4291	16	195
65	C#-4	4547	17	195
66	D-4	4817	18	209
67	D#-4	5103	19	239
68	E-4	5407	21	31
69	F-4	5728	22	96

			OSCILLATOR FREQ	
MUSICAL NOTE		Cycles/Sec.	HIGH	LOW
70	F#-4	6069	23	181
71	G-4	6430	25	30
72	G#-4	6812	26	156
73	A-4	7217	28	49
74	A#-4	7647	29	223
75	B-4	8101	31	165
80	C-5	8583	33	135
81	C#-5	9094	35	134
82	D-5	9634	37	162
83	D#-5	10207	39	223
84	E-5	10814	42	62
85	F-5	11457	44	193
86	F#-5	12139	47	107
87	G-5	12860	50	60
88	G#-5	13625	53	57
89	A-5	14435	56	99
90	A#-5	15294	59	190
91	B-5	16203	63	75
96	C-6	17167	67	15
97	C#-6	18188	71	12
98	D-6	19269	75	69
99	D#-6	20415	79	191
100	E-6	21629	84	125
101	F-6	22915	89	131
102	F#-6	24278	94	214
103	G-6	25721	100	121
104	G#-6	27251	106	115
105	A-6	28871	112	199
106	A#-6	30588	119	124
107	B-6	32407	126	151
112	C-7	34334	134	30
113	C#-7	36376	142	24
114	D-7	38539	150	139
115	D#-7	40830	159	126
116	E-7	43258	168	250
117	F-7	45830	179	6
118	F#-7	48556	189	172
119	G-7	51443	200	243
120	G#-7	54502	212	230
121	A-7	57743	225	143
122	A#-7	61176	238	248
123	B-7	64814	253	46

USTAWIANIE PARAMETRÓW FILTRÓW

ADRES	OPIS
54293	Młodszy bajt częstotliwości progowej filtru (0 - 7)
54294	Starszy bajt częstotliwości progowej filtru (0 - 255)
54295	Rezonans (bity 4 - 7) Filtrowanie kanału 3 (bit 2) Filtrowanie kanału 2 (bit 1) Filtrowanie kanału 1 (bit 0)
54296	Filtr górnoprzepustowy (bit 6) Filtr pasmowoprzepustowy (bit 5) Filtr dolnoprzepustowy (bit 4) Siła głosu (bity 0 - 3)

DODATEK N

LITERATURA

Poniżej przedstawiona jest literatura w języku angielskim pomocna przy pisaniu programów.

WYDAWCA	TYTUŁ I AUTORZY
Addison-Wesley	"BASIC and the Personal Computer", Dwyer and Critchfield
Compute	"Compute's First Book of PET/CBM"
Cowbay Computing	"Feed me, I'm Your PET Computer", Carol Alexander
	"Looking Good with Your PET", Carol Alexander
	"Teacher's PET - Plans, Quizzes, and Answers"
Creative Computing	"Getting Acquainted With Your VIC 20", T. Hartnell.
Dilithium Press	"BASIC Basic-English Dictionary for PET", Larry Noonan
	"PET BASIC", Tom Rugg and Phil Feldman.
Faulk Backer Associates	"MOS Programming Manual", MOS Technology
Hayden Book Co.	"BASIC From the Ground Up", David E. Simon
	"I Speak BASIC to My PET", Aubrey Jones, Jr.
	"Library of PET Subroutines", Nick Hampshire
	"PET Graphics", Nick Hampshire
	"BASIC Conversion Handbook, Apple, TRS-80, and PET", David A. Brain, Phillip R. Oviatt, Paul J. Paquin, and Chandler P. Stone.
Howard W. Sams	"The Howard W. Sams Crash Course in Microcomputers", Louis E. Frenzel, Jr.
	"Mostly BASIC: Applications For Your PET", Howard Berenbon
	"PET Interfacing", James M. Downey and Steven M. Rogers
	"VIC 20 Programmer's Reference Guide", A. Finkel, P. Higginbottom, N. Harris, M. Tomczyk
Level Ltd.	"Programming the PET/CBM", Raeto West
Little, Brown & Co.	"Computer Games for Businesses, Schools, and Homes", J. Victor Nagigian, William S. Hodges
	"The Computer Tutor: Learning Activities for Homes and Schools", Gary W. Orwig, University of Central Florida, William S. Hodges
McGraw-Hill	"Hands-On BASIC with a PET", Herbert D. Peckman
	"Home and Office Use of VisiCalc", D. Castlewitz, and L Chisauki
Osborne/McGraw-Hill	"PET/CBM Personal Computer Guide", Carroll S. Donahue
	"PET Fun and Games", R. Jeffries, G. Fisher
	"PET and IEEE", A. Osborne, C. Donahue
	"Some Common BASIC Programs for the PET", L. Pool, M. Borchers, and C. Donahue
	"Osborne CP/M User Guide", Thom Hogan
	"CBM Professional Computer Guide"
	"The PET Personal Computer Guide"
	"The 8086 Book", R. Rector, and G. Alexy
P. C. Publications	"Beginning Self-Teaching Computer Lessons"
Prentice-Hall	"The PET Personal Computer for Beginners", S. Dunn and V. Morgan
Reston Publishing Co .	"PET and IEEE 488 Bus (GPIB)", Eugene Fisher and C.W. Jensen
	"PET BASIC - Training Your PET Computer", Ramon Zamora, Wm. F. Carrie, and B. Albrecht
	"PET Games and Recreation", M. Ogelsby, L. Lindsley, and D. Kunkin
	"PET BASIC", R. Huskell
	"VIC Games and Recreation"
Telemas Courseware	"BASIC and the Personal Computer", T. A. Ratings, Dwyer, and M. Critchfield
Total Information Services	"Understanding Your PET/CBM, Vol. 1, BASIC Programming"
	"Understanding Your VIC", D. Schultz

Magazyny komputerowe poświęcone komputerom Commodore dostarczają najświeższych informacji o tym komputerze. Dwa magazyny, które warto zaprenumerować to:

COMMODORE - The Microcomputer Magazine, jest to dwumiesięcznik, wydawany w języku angielskim.
POWER/PLAY - The Home Computer Magazine, jest to kwartalnik wydawany w języku angielskim.

DODATEK O

MAPA REJESTRÓW SPRITE

Register # Dec	Hex	D87	D86	D85	D84	D83	D82	D81	D80	
0	0	S0X7							S0X0	SPRITE 0 X Component
1	1	S0Y7							S0Y0	SPRITE 0 Y Component
2	2	S1X7							S1X0	SPRITE 1 X
3	3	S1Y7							S1Y0	SPRITE 1 Y
4	4	S2X7							S2X0	SPRITE 2 X
5	5	S2Y7							S2Y0	SPRITE 2 Y
6	6	S3X7							S3X0	SPRITE 3 X
7	7	S3Y7							S3Y0	SPRITE 3 Y
8	8	S4X7							S4X0	SPRITE 4 X
9	9	S4Y7							S4Y0	SPRITE 4 Y
10	A	S5X7							S5X0	SPRITE 5 X
11	B	S5Y7							S5Y0	SPRITE 5 Y
12	C	S6X7							S6X0	SPRITE 6 X
13	D	S6Y7							S6Y0	SPRITE 6 Y
14	E	S7X7							S7X0	SPRITE 7 X Component
15	F	S7Y7							S7Y0	SPRITE 7 Y Component
16	10	S7X8	S6X8	S5X8	S4X8	S3X8	S2X8	S1X8	S0X8	MSB of X COORD
17	11	RC8	ECM	BMM	BLNK	RSEL	YSCL2	YSCL1	YSCL0	Y SCROLL MODE
18	12	RC7	RC6	RC5	RC4	RC3	RC2	RC1	RC0	RASTER
19	13	LPX7							LPX0	LIGHT PEN X
20	14	LPY7							LPY0	LIGHT PEN Y
21	15	SE7							SE0	SPRITE ENABLE (ON/OFF)
22	16	N.C.	N.C.	RST	MCM	CSEL	XSCL2	XSCL1	XSCL0	X SCROLL MODE
23	17	SEXY7							SEXY0	SPRITE EXPAND Y
24	18	VS13	VS12	VS11	VS10	CB13	CB12	CB11	N.C.	SCREEN CHARACTER MEMORY
25	19	IRQ	N.C.	N.C.	N.C.	LPIRQ	ISSC	ISBC	RIRQ	INTERUPT REQUEST'S
26	1A	N.C.	N.C.	N.C.	N.C.	MLPI	MISSC	MISBC	MRIRQ	INTERUPT REQUEST MASKS
27	1B	BSP7							BSP0	BACKGROUND SPRITE PRIORITY
28	1C	SCM7							SCM0	MULTICOLOR SPRITE SELECT
29	1D	SEXX7							SEXX0	SPRITE EXPAND X
30	1E	SSC7							SSC0	SPRITE-SPRITE COLLISION
31	1F	SBC7							SBC0	SPRITE-BACKGROUND COLLISION

Register # Dec	Hex	COLOR
32	20	BORDER COLOR
33	21	BACKGROUND COLOR 0
34	22	BACKGROUND COLOR 1
35	23	BACKGROUND COLOR 2
36	24	BACKGROUND COLOR 3
37	25	SPRITE MULTICOLOR 0
38	26	SPRITE MULTICOLOR 1
39	27	SPRITE 0 COLOR
40	28	SPRITE 1 COLOR
41	29	SPRITE 2 COLOR
42	2A	SPRITE 3 COLOR
43	2B	SPRITE 4 COLOR
44	2C	SPRITE 5 COLOR
45	2D	SPRITE 6 COLOR
46	2E	SPRITE 7 COLOR

Dec	Hex	COLOR
0	0	BLACK
1	1	WHITE
2	2	RED
3	3	CYAN
4	4	PURPLE
5	5	GREEN
6	6	BLUE
7	7	YELLOW
8	8	ORANGE
9	9	BROWN
10	A	LT. RED
11	B	GRAY 1
12	C	GRAY 2
13	D	LT. GREEN
14	E	LT. BLUE
15	F	GRAY 3

Uwaga: W trybie wielokolorowym (multicolor) dostępne są tylko kolory 0-7.

DODATEK P

MAPA REJESTRÓW UKŁADU 6566/6567 (VIC II)

Układy 6566/6567 są uniwersalnymi kontrolerami video, które mogę być zastosowane w komputerach i w zestawach gier telewizyjnych. Oba układy zawierają 47 rejestrów dostępnych przez standardową ośmiobitową szynę danych mikroprocesora ó5xx i mogące adresować do 16kB pamięci video. Poniżej przedstawione są rejestry tych układów.

ADDRESS	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	DESCRIPTION
00 ($00)	M0X7	M0X6	M0X5	M0X4	M0X3	M0X2	M0X1	M0X0	MOB 0 X-position
01 ($01)	M0Y7	M0Y6	M0Y5	M0Y4	M0Y3	M0Y2	M0Y1	M0Y0	M0B 0 Y-position
02 ($02)	M1X7	M1X6	M1X5	M1X4	M1X3	M1X2	M1X1	M1X0	MOB 1 X-position
03 ($03)	M1Y7	M1Y6	M1Y5	M1Y4	M1Y3	M1Y2	M1Y1	M1Y0	MOB 1 Y-position
04 ($04)	M2X7	M2X6	M2X5	M2X4	M2X3	M2X2	M2X1	M2X0	MOB 2 X-position
05 ($05)	M2Y7	M2Y6	M2Y5	M2Y4	M2Y3	M2Y2	M2Y1	M2Y0	MOB 2 Y-position
06 ($06)	M3X7	M3X6	M3X5	M3X4	M3X3	M3X2	M3X1	M3X0	MOB 3 X-position
07 ($07)	M3Y7	M3Y6	M3Y5	M3Y4	M3Y3	M3Y2	M3Y1	M3Y0	MOB 3 Y-position
08 ($08)	M4X7	M4X6	M4X5	M4X4	M4X3	M4X2	M4X1	M4X0	MOB 4 X-position
09 ($09)	M4Y7	M4Y6	M4Y5	M4Y4	M4Y3	M4Y2	M4Y1	M4Y0	MOB 4 Y-position
10 ($0A)	M5X7	M5X6	M5X5	M5X4	M5X3	M5X2	M5X1	M5X0	MOB 5 X-position
11 ($0B)	M5Y7	M5Y6	M5Y5	M5Y4	M5Y3	M5Y2	M5Y1	M5Y0	MOB 5 Y-position
12 ($0C)	M6X7	M6X6	M6X5	M6X4	M6X3	M6X2	M6X1	M6X0	MOB 6 X-position
13 ($0D)	M6Y7	M6Y6	M6Y5	M6Y4	M6Y3	M6Y2	M6Y1	M6Y0	MOB 6 Y-position
14 ($0E)	M7X7	M7X6	M7X5	M7X4	M7X3	M7X2	M7X1	M7X0	MOB 7 X-position
15 ($0F)	M7Y7	M7Y6	M7Y5	M7Y4	M7Y3	M7Y2	M7Y1	M7Y0	MOB 7 Y-position
16 ($10)	M7X8	M6X8	M5X8	M4X8	M3X8	M2X8	M1X8	M0X8	MSB of X-position
17 ($11)	RC8	ECM	BMM	DEN	RSEL	Y2	Y1	Y0	See text
18 ($12)	RC7	RC6	RC5	RC4	RC3	RC2	RC1	RC0	Raster register
19 ($13)	LPX8	LPX7	LPX6	LPX5	LPX4	LPX3	LPX2	LPX1	Light Pen X
20 ($14)	LPY7	LPY6	LPY5	LPY4	LPY3	LPY2	LPY1	LPY0	Light Pen Y
21 ($15)	M7E	M6E	M5E	M4E	M3E	M2E	M1E	M0E	MOB Enable
22 ($16)	-	-	RES	MCM	CSEL	X2	X1	X0	See text
23 ($17)	M7YE	M6YE	M5YE	M4YE	M3YE	M2YE	M1YE	M0YE	MOB Y-expand
24 ($18)	VM13	VM12	VM11	VM10	CB13	CB12	CB11	-	Memory Pointers
25 ($19)	IRQ	-	-	-	ILP	IMMC	IMBC	IRST	Interrupt Register
26 ($1A)	-	-	-	-	ELP	EMMC	EMBC	ERST	Enable Interrupt
27 ($1B)	M7DP	M6DP	M5DP	M4DP	M3DP	M2DP	M1DP	M0DP	MOB-DATA Priority
28 ($1C)	M7MC	M6MC	M5MC	M4MC	M3MC	M2MC	M1MC	M0MC	MOB Multicolor Sel
29 ($1D)	M7XE	M6XE	M5XE	M4XE	M3XE	M2XE	M1XE	M0XE	MOB X-expand
30 ($1E)	M7M	M6M	M5M	M4M	M3M	M2M	M1M	M0M	MOB-MOB Collision
31 ($1F)	M7D	M6D	M5D	M4D	M3D	M2D	M1D	M0D	MOB-DATA Collision
32 ($20)	-	-	-	-	EC3	EC2	EC1	EC0	Exterior Color
33 ($21)	-	-	-	-	B0C3	B0C2	B0C1	B0C0	Bkgd #0 Color
34 ($22)	-	-	-	-	B1C3	B1C2	B1C1	B1C0	Bkgd #1 Color
35 ($23)	-	-	-	-	B2C3	B2C2	B2C1	B2C0	Bkgd #2 Color
36 ($24)	-	-	-	-	B3C3	B3C2	B3C1	B3C0	Bkgd #3 Color
37 ($25)	-	-	. -	-	MM03	MM02	MM01	MM00	MOB Multicolor #0
38 ($26)	-	-	-	-	MM13	MM12	MM11	MM10	MOB Multicolor #1
39 ($27)	-	-	-	-	M0C3	M0C2	M0C1	M0C0	MOB 0 Color
40 ($28)	-	-	-	-	M1C3	M1C2	M1C1	M1C0	MOB 1 Color
41 ($29)	-	-	-	-	M2C3	M2C2	M2C1	M2C0	MOB 2 Color
42 ($2A)	-	-	-	-	M3C3	M3C2	M3C1	M3C0	MOB 3 Color
43 ($2B)	-	-	-	-	M4C3	M4C2	M4C1	M4C0	MOB 4 Color
44 ($2C)	-	-	-	-	M5C3	M5C2	M5C1	M5C0	MOB 5 Color
45 ($2D)	-	-	-	-	M6C3	M6C2	M6C1	M6C0	MOB 6 Color
46 ($2E)	-	-	-	-	M7C3	M7C2	M7C1	M7C0	MOB 7 Color

NOTE: A dash indicates a no connect. All no connects are read as a “1.”

DODATEK Q

USTAWIANIE PARAMETRÓW DŹWIĘKU W COMMODORE 64

Ta wygodna tabela zawiera informacje potrzebne do wykorzystania wszystkich trzech kanałów dźwięku w programach. Aby ustawić lub zmienić dany parametr należy wpisać komendę POKE, po której wpisuje się numer z drugiej kolumny, następnie przecinek i liczbę z tabeli, np.; POKE 54276,17 (Włącza generator przebiegu trójkątnego w kanale 1).

Należy pamiętać, aby ustawić poziom głosu przed wytworzeniem danego dźwięku. Poziom ten ustawia się wpisując POKE 54296 i liczbę od 0 do 15 określającą głośność wszystkich trzech kanałów.

Ustawienie wysokości dźwięku wymaga wpisania dwóch wartości do dwóch kolejnych rejestrów. Np. wpisanie POKE 54273, 33:POKE 54272, 135 powoduje uzyskanie dolnego C w kanale 1. Oczywiście wartości podane w tabeli oraz w dodatku C nie są jedynymi możliwymi wartościami, które można wpisać do tych rejestrów. Jeżeli dźwięk podany w tabeli nie brzmi czysto, można próbować wpisać inne wartości.

SETTING VOLUME - SAME FOR ALL 3 VOICES		
VOLUME CONTROL	POKE 54296	Settings range from 0 (off) to 15 (loudest)

VOICE NUMBER 1

TO CONTROL THIS SETTING	POKE THIS NUMBER	FOLLOWED BY ONE OF THESE NUMBERS (0 to 15 … or … 0 TO 255 DEPENDING on range)											
TO PLAY A NOTE		C	C#	D	D#	E	F	F#	G	G#	A	A#	B
HIGH FREQUENCY	54273	33	35	37	39	42	44	47	50	53	56	59	63
LOW FREQUENCY	54272	135	134	162	223	62	193	107	60	57	99	190	75
WAVEFORM	POKE	TRIANGLE			SAWTOOTH			PULSE			NOISE		
	54276	17			33			65			129		
PULSE RATE (Pulse Waveform)													
HI PULSE	54275	A value of 0 to 15 (for Pulse waveform only)											
LO PULSE	54274	A value of 0 to 255 (for Pulse waveform only)											
ATTACK/DECAY	POKE	ATK4	ATK3	ATK2	ATK1	DEC4	DEC3	DEC2	DEC1				
	54277	128	64	32	16	8	4	2	1				
SUSTAIN/RELEASE	POKE	SUS4	SUS3	SUS2	SUS1	REL4	REL3	REL2	REL1				
	54278	128	64	32	16	8	4	2	1				

VOICE NUMBER 2

TO PLAY A NOTE		C	C#	D	D#	E	F	F#	G	G#	A	A#	B
HIGH FREQUENCY	54280	33	35	37	39	42	44	47	50	53	56	59	63
LOW FREQUENCY	54279	135	134	162	223	62	193	107	60	57	99	190	75
WAVEFORM	POKE	TRIANGLE			SAWTOOTH			PULSE			NOISE		
	54283	17			33			65			129		
PULSE RATE (Pulse Waveform)													
HI PULSE	54282	A value of 0 to 15 (for Pulse waveform only)											
LO PULSE	54281	A value of 0 to 255 (for Pulse waveform only)											
ATTACK/DECAY	POKE	ATK4	ATK3	ATK2	ATK1	DEC4	DEC3	DEC2	DEC1				
	54284	128	64	32	16	8	4	2	1				
SUSTAIN/RELEASE	POKE	SUS4	SUS3	SUS2	SUS1	REL4	REL3	REL2	REL1				
	54285	128	64	32	16	8	4	2	1				

VOICE NUMBER 3

TO PLAY A NOTE		C	C#	D	D#	E	F	F#	G	G#	A	A#	B
HIGH FREQUENCY	54287	33	35	37	39	42	44	47	50	53	56	59	63
LOW FREQUENCY	54286	135	134	162	223	62	193	107	60	57	99	190	75
WAVEFORM	POKE	TRIANGLE			SAWTOOTH			PULSE			NOISE		
	54290	17			33			65			129		
PULSE RATE (Pulse Waveform)													
HI PULSE	54289	A value of 0 to 15 (for Pulse waveform only)											
LO PULSE	54288	A value of 0 to 255 (for Pulse waveform only)											
ATTACK/DECAY	POKE	ATK4	ATK3	ATK2	ATK1	DEC4	DEC3	DEC2	DEC1				
	54291	128	64	32	16	8	4	2	1				
SUSTAIN/RELEASE	POKE	SUS4	SUS3	SUS2	SUS1	REL4	REL3	REL2	REL1				
	54292	128	64	32	16	8	4	2	1				

PARAMETRY DO SYMULACJI RÓŻNYCH INSTRUMENTÓW

Instrument	Przebieg	Narast./Opad.	Podtrz./Wybrzm.	Szer. impulsu
Pianino	Prostokąt	9	0	St-0, Mł-25
Flet	Trójkąt	96	0	-
Klawikord	Piła	9	0	-
Ksylofon	Trójkąt	9	0	-
Organy	Trójkąt	0	240	-
Akordeon	Trójkąt	102	0	-
Trąbka	Piła	96	0	-

ZNACZENIE UŻYTYCH TERMINÓW

Narastanie	(Attack)	Prędkość narastania dźwięku do wartości szczytowej.
Opadanie	(Decay)	Prędkość opadania dźwięku od wartości szczytowej do poziomu podtrzymania.
Podtrzymanie	(Sustain)	Poziom trwającego dźwięku.
Wybrzmiewanie	(Release)	Prędkość wybrzmienia dźwięku.
Przebieg	Kształt fali dźwiękowej.	

Uwaga: Parametry obwiedni dźwięku należy wpisać przed włączeniem danego generatora fali.

DODATEK R

SPECYFIKACJA UKŁADU 6581 (SID)

Układ 6581 jest scalonym, jednoukładowym, trzykanałowym generatorem dźwięku/efektów dźwiękowych, współpracującym z rodziną mikroprocesorów 65xx. Zastosowanie takiego specjalizowanego układu upraszcza programowanie dźwięku. Układ ten można zastosować w komputerach oraz w zestawach gier telewizyjnych.

OGÓLNA CHARAKTERYSTYKA UKŁADU:

- 3 KANAŁY DŹWIĘKU
  Zakres 0 - 4 kHz.
- 4 KSZTAŁTY PRZEBIEGU
  Trójkątny, piłokształtny, prostokątny, szumy.
- 3 MODULATORY AMPLITUDY
  Zakres: 48 dB
- 3 GENERATORY OBWIEDNI
  Regulacja w skali logarytmicznej
  Prędkość narastania: 2 ms - 8 s.
  Prędkość opadania: 6 ms - 24 s.
  Poziom podtrzymania: 0 - wartość szczytowa.
  Prędkość wybrzmiewania: 6 ms - 24 s.
- SYNCHRONIZACJA OSCYLATORA
- MODULACJA KOŁOWA.

OPIS

Układ 6581 zawiera trzy kanały dźwięku, z których każdy może być używany niezależnie lub w połączeniu z pozostałymi (lub zewnętrznym źródłem dźwięku), do tworzenia bardziej złożonych dźwięków. Każdy kanał zawiera własny generator fali o zmiennym kształcie i częstotliwości, generator obwiedni oraz modulator amplitudy. Generator fali pozwala na uzyskanie danej częstotliwości dźwięku i jednego z czterech kształtów tej fali, różniących się zawartością częstotliwości harmonicznych, dzięki czemu można uzyskać różną barwę dźwięku. Generator obwiedni pozwala na różnicowanie amplitudy sygnału w czasie. Ponadto istnieje możliwość włączenia programowalnego filtru działającego na wybrany (lub wszystkie) kanał.

SID pozwala na odczytanie przez mikroprocesor zmieniających się sygnałów z trzeciego kanału. Sygnały te mogę służyć do tworzenia efektu vibrato i innych efektów. Kanał trzeci może być także wykorzystywany jako generator liczb losowych do gier. Ponadto układ SID zawiera dwa przetworniki A/C pozwalające na podłączenie potencjometrów, które mogą być wykorzystane jako "wiosełka" (paddles) do sterowania grami lub do sterowania parametrami dźwięku. SID może obsługiwać także zewnętrzne źródła dźwięku, co pozwala na zastosowanie kilku układów SID w układzie łańcuchowym lub inne połączenia do stworzenia systemu polifonicznego.

REJESTRY STERUJĄCE SID

Poniżej przedstawiono pełen spis dwudziestu dziewięciu rejestrów układu SID.

	ADDRESS					REG #	DATA								REG NAME	REG TYPE
	A_4	A_3	A_2	A_1	A_0	(HEX)	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Voice 1	
0	0	0	0	0	0	00	F_7	F_6	F_5	F_4	F_3	F_2	F_1	F_0	FREQ LO	WRITE ONLY
1	0	0	0	0	1	01	F_{15}	F_{14}	F_{13}	F_{12}	F_{11}	F_{10}	F_9	F_8	FREQ HI	WRITE ONLY
2	0	0	0	1	0	02	PW_7	PW_6	PW_5	PW_4	PW_3	PW_2	PW_1	PW_0	PW LO	WRITE ONLY
3	0	0	0	1	1	03	-	-	-	-	PW_{11}	PW_{10}	PW_9	PW_8	PW HI	WRITE ONLY
4	0	0	1	0	0	04	NOISE	⊓	◿	∧	TEST	RING MOD	SYNC	GATE	CONTROL REG	WRITE ONLY
5	0	0	1	0	1	05	ATK_3	ATK_2	ATK_1	ATK_0	DCY_3	DCY_2	DCY_1	DCY_0	ATTACK/DECAY	WRITE ONLY
6	0	0	1	1	0	06	STN_3	STN_2	STN_1	STN_0	RLS_3	RLS_2	RLS_1	RLS_0	SUSTAIN/RELEASE	WRITE ONLY

	ADDRESS					REG #	DATA								REG NAME	REG TYPE
	A_4	A_3	A_2	A_1	A_0	(HEX)	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Voice 2	
7	0	0	1	1	1	07	F_7	F_6	F_5	F_4	F_3	F_2	F_1	F_0	FREQ LO	WRITE ONLY
8	0	1	0	0	0	08	F_{15}	F_{14}	F_{13}	F_{12}	F_{11}	F_{10}	F_9	F_8	FREQ HI	WRITE ONLY
9	0	1	0	0	1	09	PW_7	PW_6	PW_5	PW_4	PW_3	PW_2	PW_1	PW_0	PW LO	WRITE ONLY
10	0	1	0	1	0	0A	-	-	-	-	PW_{11}	PW_{10}	PW_9	PW_8	PW HI	WRITE ONLY
11	0	1	0	1	1	0B	NOISE				TEST	RING MOD	SYNC	GATE	CONTROL REG	WRITE ONLY
12	0	1	1	0	0	0C	ATK_3	ATK_2	ATK_1	ATK_0	DCY_3	DCY_2	DCY_1	DCY_0	ATTACK/DECAY	WRITE ONLY
13	0	1	1	0	1	0D	STN_3	STN_2	STN_1	STN_0	RLS_3	RLS_2	RLS_1	RLS_0	SUSTAIN/RELEASE	WRITE ONLY

	ADDRESS					REG #	DATA								REG NAME	REG TYPE
	A_4	A_3	A_2	A_1	A_0	(HEX)	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Voice 3	
14	0	1	1	1	0	0E	F_7	F_6	F_5	F_4	F_3	F_2	F_1	F_0	FREQ LO	WRITE ONLY
15	0	1	1	1	1	0F	F_{15}	F_{14}	F_{13}	F_{12}	F_{11}	F_{10}	F_9	F_8	FREQ HI	WRITE ONLY
16	1	0	0	0	0	10	PW_7	PW_6	PW_5	PW_4	PW_3	PW_2	PW_1	PW_0	PW LO	WRITE ONLY
17	1	0	0	0	1	11	-	-	-	-	PW_{11}	PW_{10}	PW_9	PW_8	PW HI	WRITE ONLY
18	1	0	0	1	0	12	NOISE				TEST	RING MOD	SYNC	GATE	CONTROL REG	WRITE ONLY
19	1	0	0	1	1	13	ATK_3	ATK_2	ATK_1	ATK_0	DCY_3	DCY_2	DCY_1	DCY_0	ATTACK/DECAY	WRITE ONLY
20	1	0	1	0	0	14	STN_3	STN_2	STN_1	STN_0	RLS_3	RLS_2	RLS_1	RLS_0	SUSTAIN/RELEASE	WRITE ONLY

	ADDRESS					REG #	DATA								REG NAME	REG TYPE
	A_4	A_3	A_2	A_1	A_0	(HEX)	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Filter	
21	1	0	1	0	1	15	-	-	-	-	-	FC_2	FC_1	FC_0	FC LO	WRITE ONLY
22	1	0	1	1	0	16	FC_{10}	FC_9	FC_8	FC_7	FC_6	FC_5	FC_4	FC_3	FC HI	WRITE ONLY
23	1	0	1	1	1	17	RES_3	RES_2	RES_1	RES_0	FILTEX	FILT3	FILT2	FILT1	RES/FILT	WRITE ONLY
24	1	1	0	0	0	18	3 OFF	HP	BP	LP	VOL_3	VOL_2	VOL_1	VOL_0	MODE/VOL	WRITE ONLY

	ADDRESS					REG #	DATA								REG NAME	REG TYPE
	A_4	A_3	A_2	A_1	A_0	(HEX)	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Misc.	
25	1	1	0	0	1	19	PX_7	PX_6	PX_5	PX_4	PX_3	PX_2	PX_1	PX_0	POT X	READ ONLY
26	1	1	0	1	0	1A	PY_7	PY_6	PY_5	PY_4	PY_3	PY_2	PY_1	PY_0	POT Y	READ ONLY
27	1	1	0	1	1	1B	O_7	O_6	O_5	O_4	O_3	O_2	O_1	O_0	OSC_3/RANDOM	READ ONLY
28	1	1	1	0	0	1C	E_7	E_6	E_5	E_4	E_3	E_2	E_1	E_0	ENV_3	READ ONLY

DODATEK S

KOMENDY OBSŁUGUJĄCE STACJĘ DYSKÓW I DRUKARKĘ

Poniżej przedstawione są komendy pozwalające na dokonywanie różnych operacji na dyskach i drukarkach Commodore.

CLOSE

FORMAT: CLOSE <numer zbioru>

Działanie: Komenda ta zamyka zbiór otwarty komendę OPEN. Numer zbioru musi być ten sam, co podczas otwierania zbioru komendą OPEN i przesyłania danych komendami PRINT# lub INPUT#. Jeżeli jest to zbiór na dysku komenda CLOSE powoduje zapisanie zawartości buforów stacji na dysk. Nie zamknięcie zbioru może spowodować, że zbiór na dysku będzie niekompletny, lub wręcz niemożliwy do odczytania. Przy korzystaniu z innych urządzeń zamknięcie nie jest konieczne lecz w ten sposób zwalnia się część pamięci dla pozostałych funkcji programu.

PRZYKŁADY UŻYCIA KOMENDY CLOSE:

```
10 CLOSE 1
20 CLOSE X
30 CLOSE 9*(J+1)
```

CMD

FORMAT: CMD <numer zbioru> [,łańcuch]

Działanie: Komenda ta powoduje skierowanie danych wysyłanych na ekran do określonego zbioru. Może to być zbiór na dysku, drukarka lub modem. Numer zbioru musi być przedtem określony komendą OPEN. Jeżeli po komendzie CMD podany jest łańcuch znaków, jest on wysyłany do zbioru. Można w ten sposób podać od razu np. tytuł wydruku itp.

Po wykonaniu tej komendy wszystkie rozkazy PRINT czy LIST będą wysyłały napisy w takim samym formacie do zbioru, a nie na ekran.

Aby spowodować ponowne wyświetlanie znaków na ekranie należy wysłać pustą linię komendą PRINT# do zbioru przed jego zamknięciem komendą CLOSE. Powoduje to, że urządzenie nie oczekuje już na więcej danych.

Każdy błąd (np. ?SYNTAX ERROR) powoduje skierowanie napisów ponownie na ekran. Nie zamyka on jednak zbioru, powinno się zatem po tym błędzie wykonać komendę PRINT# do zbioru i zamknąć go komendą CLOSE (Więcej informacji można znaleźć w instrukcji obsługi danego urządzenia).

PRZYKŁADY UŻYCIA KOMENDY CMD:

```
OPEN 4,4: CMD 4,"TYTUL":LIST        : REM WYDRUKOWANIE PROGRAMU NA DRUKARCE
PRINT #4: CLOSE 4                   : REM PRZYWROCENIE WYSWIETLANIA NA EKRANIE

10 OPEN 1,8,4,"TEST":REM UTWORZENIE ZBIORU SEQ NA DYSKU
20 CMD 8: REM WYSYLANIE NAPISOW NA DYSK, A NIE NA EKRAN
30 FOR L=1 TO 100
40 PRINT L:REM WPISANIE LICZBY DO BUFORU DYSKU
50 NEXT
60 PRINT #1: REM PRZYWROCENIE WYSWIETLANIA NA EKRANIE
70 CLOSE 1: REM ZAPISANIE DANYCH Z BUFORU NA DYSK
```

GET#

FORMAT: GET# <numer zbioru>, <lista zmiennych>

Działanie: Komenda GET# odczytuje ze zbioru po jednym znaku i przypisuje te znaki kolejnym zmiennym z listy. Działa tak samo jak komenda GET, tylko, że pozwala na wczytywanie z innych urządzeń niż klawiatura. Jeżeli nie odczytano żadnego znaku, zmiennej łańcuchowej jest przypisywany łańcuch pusty "", a zmiennej liczbowej 0. Komendą tą można także wczytywać znaki oddzielające dane w zbiorach, takie jak przecinek, czy kod klawisza RETURN (CHR$(13)).

Jeżeli zbiór skierowany jest na ekran (urządzenie numer 3), komenda GET# odczytuje znak wskazywany przez kursor i przesuwa kursor o jedno pole. Na końcu linii logicznej na ekranie wysyłany jest kod klawisza RETURN.

PRZYKŁADY UŻYCIA KOMENDY GET#:

```
GET# 1, A$
OPEN 1, 3: GET# 1, Z7$
GET# 1, A, B, C$, D$
```

INPUT#

FORMAT: INPUT# <numer zbioru>, <lista zmiennych>

Działanie: Jest to zazwyczaj najszybszy i najprostszy sposób na odczytywanie danych zapisanych na dysku. Dane wczytuje się w postaci całych zmiennych o długości do 80 znaków, w odróżnieniu od komendy GET#, która wczytuje tylko po jednym znaku na raz.

Komenda INPUT# uznaje, że pojedyncza zmienna to ciąg znaków zakończony jednym z separatorów: przecinek (,), średnik (;),dwukropek (:) lub kod klawisza RETURN (CHR$(13)). Jeżeli jeden z tych znaków ma być wpisany w zmienną, należy całe wyrażenie ująć w cudzysłów podczas zapisywania na dysk (patrz komenda PRINT#).

Jeżeli zmienna podana po INPUT# jest typu numerycznego, a ze zbioru zostaną wczytane inne znaki pojawi się komunikat błędu BAD DATA. Jeżeli łańcuch wczytany ze zbioru ma więcej niż 80 znaków pojawi się komunikat STRING TOO LONG.

Jeżeli zbiór skierowany jest na ekran (urządzenie 3), wczytana zostanie cała linia logiczna, a kursor przesunie się do następnej linii.

PRZYKŁADY UŻYCIA KOMENDY INPUT#:

```
10 INPUT# 1, A
20 INPUT# 1, B$, C
```

LOAD

FORMAT: LOAD "<nazwa zbioru>" , <urządzenie> [,parametr]

Działanie: Komenda ta wczytuje zawartość programu z dysku do pamięci. W ten sposób można wielokrotnie korzystać z raz napisanego programu. Zazwyczaj stacja dysków jest urządzeniem numer 8. Komenda LOAD zamyka wszystkie otwarte zbiory, a ponadto jeżeli użyta jest w trybie bezpośrednim, wykonuje komendę CLR przed wczytaniem programu. Jeżeli LOAD było wpisane w programie, po wczytaniu nowego programu automatycznie wykonywane jest polecenie RUN. Oznacza to, że można używać komendy LOAD do łączenia wielu programów. Żadne zmienne nie przepadają podczas takiego łączenia programów.

Można podać całą nazwę programu, który ma zostać wczytany, lub tylko gwiazdkę ujętą w cudzysłów ("*"), co spowoduje wczytanie pierwszego programu na dysku. Jeżeli zbiór o tej nazwie nie zostanie znaleziony lub nie jest to program (PRG), wyświetlony będzie komunikat ?FILE NOT FOUND.

Użycie dodatkowego parametru 1 spowoduje wczytanie programu w to miejsce pamięci, z którego był on zapisywany na dysk.

PRZYKŁADY UŻYCIA KOMENDY LOAD:

```
LOAD A$,8            : REM WCZYTANIE PROGRAMU O NAZWIE ZAWARTEJ W A$
LOAD "*",8           : REM WCZYTANIE PIERWSZEGO PROGRAMU NA DYSKU
LOAD "$",8           : REM WCZYTANIE KATALOGU DYSKU
LOAD "GRA",8         : REM WCZYTANIE PROGRAMU Z DYSKU
LOAD "GRA DWA",8,1   : REM WCZYTANIE PROGRAMU W OKRESLONE MIEJSCE W PAMIECI, Z KTOREGO BYL ON
                       ZAPISANY
```

OPEN

FORMAT: OPEN <num. zbioru> , <num. urządzenia> , [<parametr>] ,"<nazwa zbioru> [,<typ>] [,<tryb>] "]

Działanie: Komenda ta otwiera kanał dostępu do urządzenia peryferyjnego, którym przesyłane są informacje do/z komputera.

Nie zawsze trzeba podawać wszystkie możliwe parametry komendy OPEN. Zawsze wymagane są dwa pierwsze:

1) LOGICZNY NUMER ZBIORU
2) NUMER URZĄDZENIA

Numer zbioru jest liczbą określającą ten zbiór w komendach takich jak: PRINT#, INPUT#, GET#, CMD czy CLOSE. Liczbą tą może być dowolnie wybierana przez programistę liczba z zakresu 1 - 255.

UWAGA: Numery zbiorów większe od 128 były pierwotnie zaprojektowane do innych celów, tak więc wskazane jest używanie jako numerów zbiorów liczb poniżej 127.

Każde urządzenie peryferyjne ma swój własny numer. Parametr <numer urządzenia> określa, do lub z którego urządzenia mają być przesyłane dane. Urządzenia takie jak stacja dysków czy drukarka wymagają ponadto podania dodatkowego parametru. Jest to specjalny kod informujący urządzenie jakie działanie będzie podjęte.

Nazwa zbioru może zostać pominięta, jednakże później nie ma możliwości jej podania.

Jeżeli otwierany jest zbiór w stacji dysków <parametr> od 1 do 14 określa, że przesyłane będą dane z/do zbioru na dysku. Wartość 15 tego parametru otwiera specjalny dostęp do tzw. kanału poleceń, którym są przesyłane nie dane, ale polecenia wykonania pewnych operacji przez stację dysków (zmiana nazwy, kasowanie lub kopiowanie zbiorów). (Więcej informacji na ten temat można znaleźć w instrukcji obsługi stacji dysków).

<Nazwa programu> jest ciągiem 1 - 16 znaków i nie jest wymagana dla zbiorów na drukarce. Jeżeli <typ> nie jest podany, przyjmowany jest domyślnie program (PRG). Jeżeli nie podano <trybu>, to zbiory o dostępie sekwencyjnym (SEQ) są domyślnie przyjmowane jako otwarte do czytania (tryb = R (ang. Read)). Można określić tryb zapisywania do zbioru podając literę W (ang. Write).

Próba dostępu komendą PRINT#, GET# itp. do zbioru, który nie był przedtem otwarty komendą OPEN, spowoduje wyświetlenie komunikatu **?FILE NOT OPEN**. Próba otworzenia do czytania (tryb R) zbioru, który nie istnieje na dysku spowoduje pojawienie się błędu **?FILE NOT FOUND**. Natomiast otwieranie do zapisu (tryb W) zbioru, który już istnieje na dysku spowoduje błąd **?FILE EXISTS**. Próba otworzenia zbioru już przedtem otwartego i nie zamkniętego komendą CLOSE spowoduje błąd **?FILE OPEN**.

PRZYKŁADY UŻYCIA KOMENDY OPEN:

```
OPEN 2,8,4,"ZAPIS,SEQ,W"        : REM otwiera zbiór sekwencyjny do zapisywania danych na dysku
OPEN 50,0                       : REM odczyt z klawiatury
OPEN 12,3                       : REM zapis na ekran
OPEN 17,4                       : REM zapis na drukarkę
OPEN 1,2,0,CHR$(10)             : REM otwarcie kanału dla RS-232 (MODEM)
OPEN 1,4,0,"NAPIS"              : REM wybranie zestawu dużych znaków na drukarce
OPEN 1,4,7,"NAPIS"              : REM wybranie zestawu małych znaków na drukarce
OPEN 1,5,7,"NAPIS"              : REM wybranie zestawu małych znaków na drukarce o numerze urządzenia 5
OPEN 1,8,15,"POLECENIE"         : REM wysłanie polecenia do stacji dysków
OPEN 1,8,1,"NAZWA,L"+CHR$(X)    : REM utworzenie zbioru o dostępie swobodnym (REL) o długości rekordu X.
```

PRINT#

FORMAT: PRINT#<numer zbioru> [<zmienna>][<,;> <zmienna>] ...

Działanie: Komenda PRINT# zapisuje dane w zbiorze logicznym. Numer zbioru musi być ten sam, co w komendzie OPEN. Dane wysyłane są do urządzenia, którego numer podany był w instrukcji OPEN. Wyrażenie <zmienna> może być dowolnego rodzaju. Znaki oddzielające kolejne zmienne (separatory) są takie same, jak w przypadku komendy PRINT. Działanie ich jest nieco inne w dwóch kwestiach.

Jeżeli na końcu listy nie występuje żaden separator, to wysyłany jest kod klawisza RETURN CHR$(13). Jeżeli lista kończyła się separatorem (obojętnie przecinkiem czy średnikiem), kod CHR$(13) nie jest wysyłany. Niezależnie od użytego separatora następna zmienna wysyłana jest bezpośrednio po poprzedniej. Kod CHR$(13) powoduje, że instrukcja INPUT# przestaje wczytywać dane i wszystkim pozostałym zmiennym z listy przypisane są wartości 0 lub puste łańcuchy znaków zależnie od ich typów. Dane znajdujące się za kodem CHR$(13) będą wczytane kolejną instrukcję INPUT#.

Najprostszym sposobem na wysłanie kilku linii danych jednym poleceniem PRINT#, jest przypisać jakiejś zmiennej wartość CHR$(13) i zmienną tę oddzielać wyrażenia, które maję się znaleźć w kolejnych liniach.

PRZYKŁADY UŻYCIA KOMENDY PRINT#;

1)
```
10 OPEN 1,8,4,"PRZYKLAD"
20 R$=CHR$(13)
30 PRINT# 1,1;R$;2;R$;3;R$;4;R$;5
40 PRINT# 1,6
50 PRINT# 1,7
```

2)
```
10 CO$=CHR$(44):CR$=CHR$(13)
20 PRINT# 1,"AA"CO$"BB","CC";"DD";CR$;"EE";CR$
30 INPUT# 1, A$, BCD$, E$
```

3)
```
5 CR$=CHR$(13)
10 PRINT# 2,"AAA";CR$;"BBB";CR$
20 PRINT# 2, "CCC";
30 INPUT# 2, A$,B$,PUSTY$,C$
```

SAVE

FORMAT: SAVE "<nazwa zbioru>" , <numer urządzenia> [, <parametr>]

Działanie: Komendą SAVE zapisuje się program znajdujący się w pamięci komputera na dysk lub magnetofon. Program po zapisaniu na dyskietkę nadal pozostaje w pamięci komputera. Na dyskietce program przechowywany jest w postaci zbioru typu PRG. Komenda SAVE może znajdować się w linii programu. Po zapisaniu programu wykonywana będzie następna instrukcja programu.

Przy zapisywaniu programu na dysk konieczne jest podanie nazwy.

PRZYKŁADY UŻYCIA KOMENDY SAVE:
SAVE "PROGRAM",8
SAVE A$,8

ZAPISANIE NOWSZEJ WERSJI PROGRAMU

FORMAT: SAVE "@0:<nazwa zbioru>", <numer urządzenia>

Działanie: Takie użycie komendy SAVE powoduje zapisanie programu pod nazwą już istniejącą na dysku. Normalne użycie komendy SAVE wymaga, aby zbiór o podanej nazwie nie istniał na dysku, w przeciwnym wypadku program nie zostanie zapisany, pomimo, że nie pojawi się komunikat o błędzie.

PRZYKŁADY UŻYCIA KOMENDY
SAVE"@0:MOJ PROGRAM",8: REM Powoduje usunięcie z dysku poprzedniej wersji programu i zapisanie nowej.

VERIFY

FORMAT: VERIFY "<nazwa zbioru>" , <numer urządzenia>

Działanie: Komenda VERIFY służy do sprawdzenia poprawności ostatniego zapisu programu na dysku. Zazwyczaj używa się jej bezpośrednio po komendzie SAVE.
Przy sprawdzaniu programu na dysku, musi być podana nazwa programu. Jeżeli program w pamięci i na dysku różnią się między sobą, pojawi się komunikat **?VERIFY ERROR**.

PRZYKŁADY UŻYCIA KOMENDY VERIFY:
9000 SAVE "PROGRAM",8
9001 VERIFY "PROGRAM",8

Materiał pobrany ze strony magazynu "Filety & 64NET" http://www.filety.prv.pl/.